Repousser les limites d'Access - récupérer un fil RSS

par Charles A.

Date de publication : 01/10/2005

Dernière mise à jour : 01/10/2005

Thèmes abordés : . récupération de source http . parsing xml via une classe . import de données Niveau requis : confirmé

I - Introduction

Ce tutoriel cherche à repousser les limites d'utilisation fréquente de Microsoft Access.

A travers la fonctionnalité étudiée : la récupération d'un fil RSS de blog, il permet d'aborder ces différents thèmes :

. récupération d'une source HTTP par deux méthodes

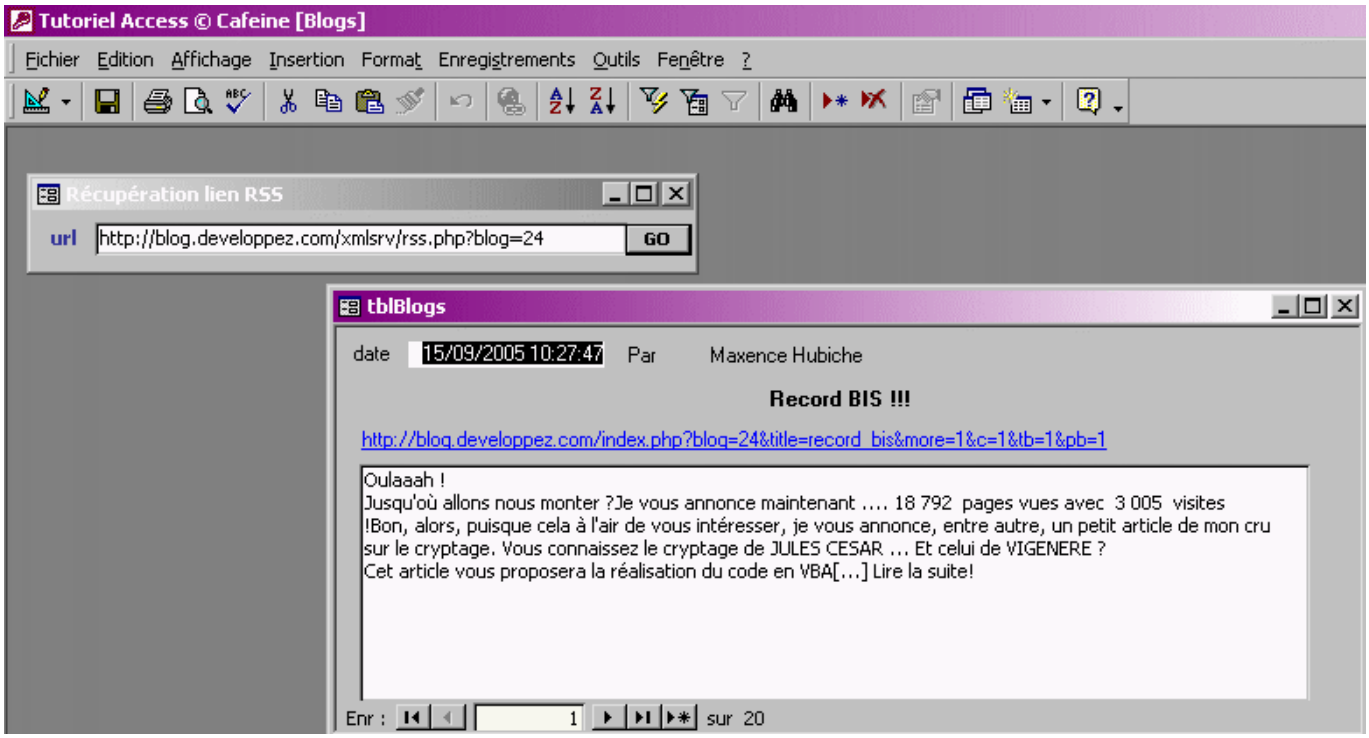
. travail sur les chaines

. parsing XML

. utilisation d'une classe

. ajout de données

II - Résultat final et méthode



Par un simple clic sur le bouton "GO", l'application Access permet de visualiser dans un formulaire une synthèse du blog.

Si nous décomposons le fonctionnement :

. récupération du lien du fil RSS du blog

. enregistrement dans un fichier local

. chargement du fichier local au format XML

. parsing XML du fichier

. intégration des informations dans une table.

III - Récupération du fil RSS du blog

A - Le Weblog et sa syndication RSS

> ***RSS**, sigle de **Really Simple Syndication** (syndication vraiment simple), ou de **Rich Site Summary** (résumé complet d'un site) est un format de syndication de contenu Web. C'est un dialecte de XML. Il existe sept formats différents de RSS, ce qui rend indispensable l'établissement d'une norme.*
>
> *Il est à noter que Syndicate, en anglais, est en rapport avec le journalisme et la vente d'un article à plusieurs journaux. Really Simple Syndication se rapproche donc d'une diffusion journalistique simplifiée.*
>
> *Pour plus d'informations cliquez ici.*

Pour lire un Weblog (blog) nous allons exploiter la fonctionnalité de *syndication* présente sur la plupart de ces sites pour s'affranchir du formatage ou de la présentation qu'un format HTML rendrait incompatible d'un blog à un autre.

Cette syndication est en fait un lien, qui pointe sur un fichier au format XML, normé pour ne présenter que l'information résumée en otant tout enrichissement de forme.

Les blogs de developpez.com utilisent 4 normes de syndication, nous en avons choisi une simple : la norme RSS 0.92 (userland).

Vous pouvez regarder à quoi ressemble un fichier RSS en cliquant ici.

Nous parcourons maintenant le blog de Maxence Hubiche, et nous récupérons le lien de sa syndication RSS 0.92 :

http://blog.developpez.com/xmlsrv/rss.php?blog=24

C'est sur ce fichier que nous allons travailler, pour cela nous allons le récupérer.

B - Récupérer une source par le protocole HTTP - méthode API wininet

```
Option Compare Database
Option Explicit


'Fonction pour ouvrir une connexion Internet :
Public Declare Function InternetOpen Lib "wininet.dll" Alias "InternetOpenA" _
        (ByVal lpszAgent As String, ByVal dwAccessType As Long, ByVal lpszProxyName As String, _
         ByVal lpszProxyBypass As String, ByVal dwFlags As Long) As Long

'Fonction pour fermer le handle de la connexion :
```

```
Public Declare Function InternetCloseHandle Lib "wininet.dll" (ByVal hInet As Long) As Integer

'Fonction pour ouvrir une adresse URL :
Public Declare Function InternetOpenUrl Lib "wininet.dll" Alias "InternetOpenUrlA" _
        (ByVal hInternetSession As Long, ByVal lpszUrl As String, ByVal lpszHeaders As String, _
         ByVal dwHeadersLength As Long, ByVal dwFlags As Long, ByVal dwContext As Long) As Long

'Fonction pour lire les données d'une URL :
Public Declare Function InternetReadFile Lib "wininet.dll" _
        (ByVal hFile As Long, ByVal lpBuffer As String, ByVal dwNumberOfBytesToRead As Long, _
         lNumberOfBytesRead As Long) As Integer

Public Const INTERNET_OPEN_TYPE_PRECONFIG = 0   ' utiliser info de config de la base de registre
Public Const INTERNET_FLAG_EXISITING_CONNECT = &H20000000
Public Const INTERNET_FLAG_RELOAD = &H80000000

Private Declare Function GetTickCount Lib "kernel32" () As Long
Private total As Long
Private Nb As Long
Dim hSession As Long

Public Sub openInter()
    hSession = InternetOpen("MonApp", INTERNET_OPEN_TYPE_PRECONFIG, vbNullString, vbNullString, 0)
    total = 0
    Nb = 0
End Sub

Public Sub closeInter()
    InternetCloseHandle (hSession)
End Sub


Public Sub GetWeblog(ByVal url As String, ByVal strFic As String)

    ' pointeur du lien lien
    Dim hUrlFile As Long
    Dim bBoucle As Boolean
    ' bloc de lecture par buffer 4 096 caractères
    Dim sReadBuf As String * 4096
    Dim OctetsLus As Long
    ' pointeurs des fichiers
    Dim localFile As Long
    ' chronométrage du temps d'exécution
    Dim t0 As Single, t1 As Single

    t0 = GetTickCount()

    ' ouverture des ressources de navigation internet
    openInter

    ' désignation d'un pointeur de fichier libre
    localFile = FreeFile

    ' si le fichier existe déjà on l'efface
    If Len(Dir(strFic)) > 0 Then Kill strFic

    ' ouverture du fichier en mode binaire
    Open strFic For Binary As #localFile

    ' ouverture de l'url
    hUrlFile = InternetOpenUrl(hSession, url, vbNullString, 0, INTERNET_FLAG_RELOAD, 0)

    bBoucle = True
    While bBoucle
        sReadBuf = ""
        ' lecture par bloc de 4096 caractères
        bBoucle = InternetReadFile(hUrlFile, sReadBuf, 4096&, OctetsLus)
        ' écriture par bloc dans le fichier local
        Put #localFile, , Left(sReadBuf, OctetsLus)
        If OctetsLus = 0 Then bBoucle = False
        DoEvents
    Wend
    ' fermeture du fichier local
    Close #localFile
    ' fermeture des ressources de navigation
    closeInter

    t1 = GetTickCount()
    Debug.Print "téléchargement du xml : "; Format((t1 - t0) / 1000, "0.000") & " s"
```

```
    ' parsing XML du fichier nous verrons ce point plus tard dans l'article
    xmlParser strFic

End Sub
```

Ce code utilise les API wininet pour charger en mémoire une ressource du protocole HTTP.

Nous l'écrivons dans un fichier local par un mode d'accès binaire par blocs de 4096 caractères.

NB : le temps mis à récupérer ce fichier est affiché dans la fenêtre d'exécution : 1,388 s

C - Récupérer une source par le protocole HTTP - méthode de la bibliothèque XML

```
Function GetWeblog2(ByVal LinkHTTP As String, ByVal IntoFile As String) As String
'-------------------------------------------------------------------------------------
' Procedure native : GetHTML
' Créée le  : vendredi 27 mai 2005 15:15
' Auteur    : Maxence
' Objet     : Récupère le texte HTML d'une page Web
' Bibliothèque : Cette procédure nécessite la déclaration de la bibliothèque
'                'Microsoft XML v x.xx' - prenez la version la plus récente
'-------------------------------------------------------------------------------------
'
' getweblog2 "http://blog.developpez.com/xmlsrv/rss.php?blog=24","D:\Developpez\Access\blog.rss"
'
    'Définition des variables
      Dim oHttp As MSXML2.ServerXMLHTTP40
      Dim sTemp As String
      Dim nLimite As Long

      'Définition des constantes
      Const conStatutOK As Long = 200


    Dim t0 As Single, t1 As Single
    Dim localFile As Long

    t0 = GetTickCount()

    localFile = FreeFile
    If Len(Dir(IntoFile)) > 0 Then Kill IntoFile
    Open IntoFile For Binary As #localFile

      ' Instancier l'objet
      Set oHttp = New MSXML2.ServerXMLHTTP40

      With oHttp
        ' Se Connecter à la page web et récupérer l'information.
        .Open "GET", LinkHTTP, False
        .Send
       ' v = .getAllResponseHeaders
        sTemp = .responseText
        'Vérifier que tout s'est bien passé
        If Not .Status = conStatutOK Then Err.Raise 65000, "Procédure", "pas trouvé !"
      End With

    On Error Resume Next
    oHttp.abort
    Set oHttp = Nothing


    Put #localFile, , sTemp
    Close #localFile

    t1 = GetTickCount()
    Debug.Print "téléchargement du xml : "; Format((t1 - t0) / 1000, "0.000") & " s"
```

```
        xmlParser IntoFile

    End Function
```

Je me suis inspiré d'une source de Maxence Hubiche (ça tombe bien).

Ce code nécessite la déclaration d'une référence **Microsoft XML X.XX**

NB : le temps de récupération est identique.

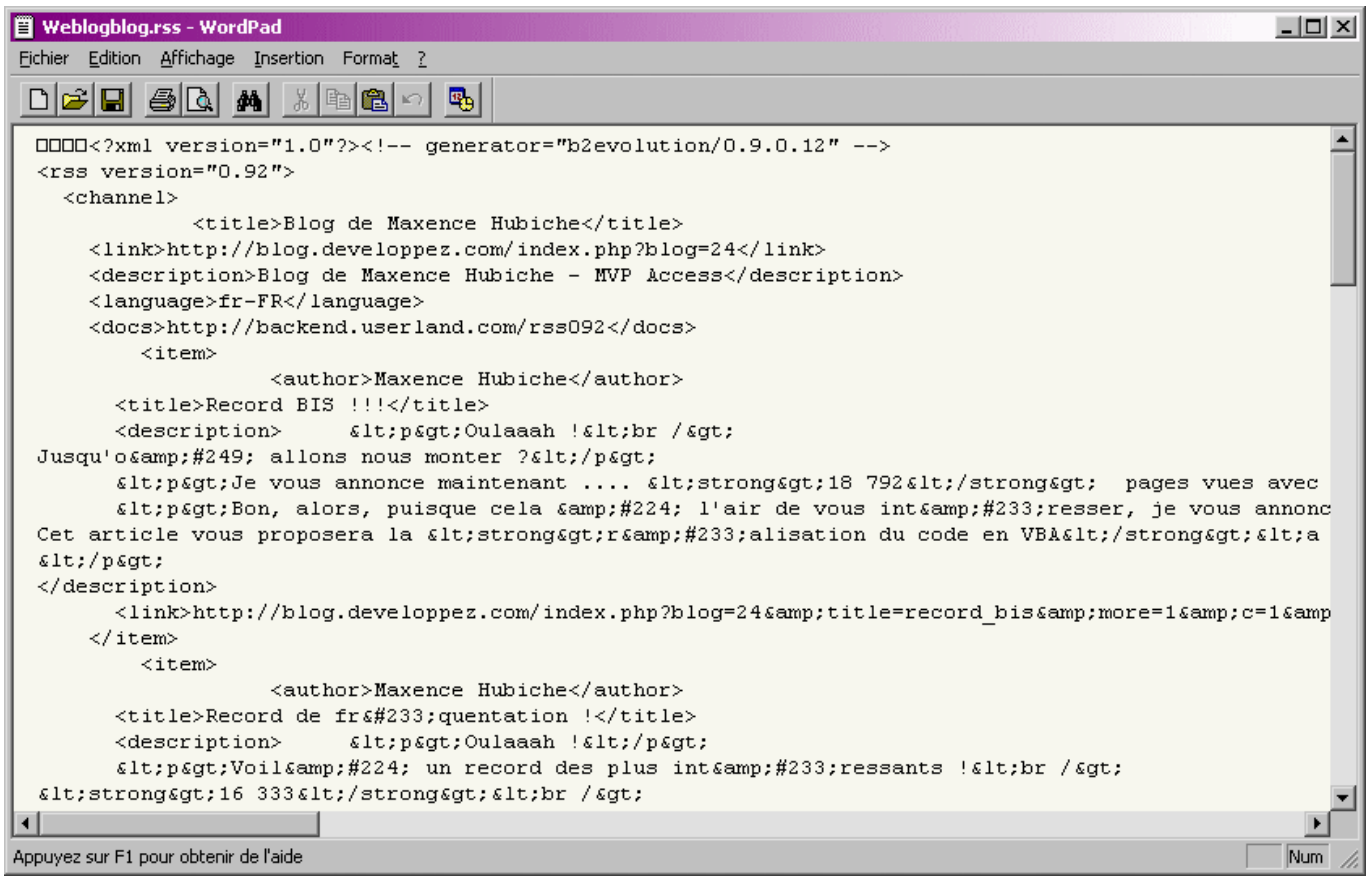
J'aurais cependant tendance à ne pas trop conseiller cette méthode en raison de la limite en taille d'une variable de type String (64 ko).

La méthode "bufferisée" permet de ne pas être limité quant à la taille de la source HTTP.

IV - Parser le fichier local XML

A - Analyser la structure

Voici ce que nous trouvons à l'ouverture du fichier source :



A la lecture attentive nous remarquons que la structure suivante se répète :

```
<item>
  <author>Nom de l'auteur<author>
  <title>titre de l'article</title>
  <description>blabla de l'article (pardon Maxence)</description>
  <link>url pour pointer sur l'article en version complète</link>
</item>
```

Nous allons bâtir une fonction pour convertir ce XML en une donnée intelligible pour Access.

Ce processus s'appelle le **Parsing**.

A des fins didactiques, ce tutoriel fait le choix de "parser" ce fichier sans recours à une bibliothèque extérieure, mais en écrivant un ensemble de fonctions pour traiter les informations.

Ces traitements vont être réalisés ici grâce à une **classe**.

Pour plus d'informations sur les classes, notamment celles qui manipulent des chaînes de caractères, je vous renvoie à un autre de mes tutoriels : http://cafeine.developpez.com/classe/.

B - la Classe clsToken

```
Option Compare Database

' ------------------------------------------------------
' @Auteur   :   cafeine
' @Date     :   09-09-2995
' ------------------------------------------------------
' @Project  :   clsToken
' ------------------------------------------------------
' @Desc     :   Classe permettant d'émuler des Tokens
'
'               GetTok  :   Obtenir un token selon sa position
'               DelTok  :   Supprimer un token selon sa position
'               getNextXMLnode : récupérer la chaine contenue entre
'                                <tag> et </tag>
' ------------------------------------------------------

Private classStringValue As String

Private Sub Class_Initialize()
    classStringValue = vbNullString
End Sub

Public Property Let Value(str As String)
    classStringValue = str
End Property

Public Property Get Value() As String
    Value = classStringValue
End Property

Public Function getNextXMLnode(ByVal strIdentity As String) As String

    Dim strIdentStart As String, strIdentEnd As String

    strIdentStart = "<" & strIdentity & ">"
    strIdentEnd = "</" & strIdentity & ">"

    ' effacement de ce qui précède le tag d'ouverture
    DelTok 0, strIdentStart

    ' récupération de ce qui est contenu dans le tag
    getNextXMLnode = GetTok(0, strIdentEnd)

    ' effacement de ce qui précède le tag de fermeture
    DelTok 0, strIdentEnd

End Function

Public Function DelTok(intIdToken As Integer, strDelim As String)

    Dim temp() As String

    ' découpage de la chaine en tableau de valeur de base 0
    temp = Split(classStringValue, strDelim)

    ' test de dépassement de capacité
    If intIdToken <= UBound(temp) Then
        ' effacement de la valeur définie
        classStringValue = Replace(classStringValue, GetTok(intIdToken, strDelim) + _
```

```
                                                    IIf(intIdToken < UBound(temp), strDelim,
vbNullString), _
                                                    vbNullString, _
                                                    1, _
                                                    1)

    Else
        ' on ne fait rien la valeur de la classe restera inchangée
    End If

End Function

Public Function GetTok(intIdToken As Integer, strDelim As String) As String

    Dim temp() As String

    ' découpage de la chaine en tableau de valeur de base 0
    temp = Split(classStringValue, strDelim)

    ' test de dépassement de capacité
    If intIdToken <= UBound(temp) Then
        ' renvoi de la valeur correspondante
        If InStr(classStringValue, strDelim) Then
            GetTok = temp(intIdToken)
        Else
            GetTok = vbNullString
        End If
    Else
        GetTok = vbNullString
    End If

End Function
```

Explications

Le **Token** est une notion de programmation qui est présente dans de nombreux langages (comme par exemple en JAVA avec StringTokenizer), mais qui fait hélas défaut au VB / VBA.

Nous allons remédier à cette lacune en créant un classe qui aura pour valeur une chaîne, et qui contiendra des méthodes de Token.

Cette classe contient les fonctions de Token les plus élémentaires :

. **GetTok** (Get Token) : pour une chaîne "Coucou c'est moi" GetTok(2, " ") renvoie "c'est"

. **DelTok** (Delete Token) : pour une chaîne "Coucou c'est moi" DelTok(2, " ") change la valeur en "Coucou moi".

La fonction **getNextXMLnode** permet d'isoler ce qui est contenu entre deux balises XML ouvrante et fermante (<balise> et </balise>)

Elle opère un DelTok de ce qui est avant la balise ouvrante, et un GetTok de ce qui est avant la balise fermante.

C - La fonction de Parsing

```
' Type de données pouvant recueillir un item de fil RSS
Type RSSitem_
        author As String
        title As String
        description As String
        link As String
End Type

' Encapsulation d'un tableau de RSSitem_ dans un type RSSTable
Type RSSTable_
        ItemCount As Integer
        Items() As RSSitem_
End Type

Public rss As RSSTable_

Public Function xmlParser(ByVal strFile As String)
    ' pointeur de fichier
    Dim xmlFile As Integer
    ' Buffer de chaine de caractère lecture par bloc de 512 caractères
    Dim strBuffer As String * 512
    ' Chaine servant à trouver les "item"
    Dim strItemize As String
    ' Chaines temporaires pour effectuer des opérations de texte
    Dim strTemp As String
    Dim strTempBuffer As String
    ' nous allons utiliser la classe clsToken
    '  et sa fonction évoluée : GetNextXMLnode(Nomd'unNoeud)
    Dim ssTemp As New clsToken
    Dim i As Long
    Dim iCount As Long


    Dim t0 As Single, t1 As Single

    t0 = GetTickCount()


    Erase rss.Items
    rss.ItemCount = 0

    xmlFile = FreeFile

    ' ouverture du fichier XML en lecture
    Open strFile For Binary Access Read As #xmlFile

    strItemize = vbNullString
    iCount = 0
    Do While Not EOF(xmlFile)
        ' lecture d'un bloc de 512 caractères
        Get #xmlFile, , strBuffer
        ' concaténation à la chaîne strItemize
        strItemize = strItemize + strBuffer

        ' tentative de lecture d'un node "item"
        ssTemp.Value = strItemize
        strTemp = ssTemp.getNextXMLnode("item")

        ' si le node "item" a été trouvé
        Do While Len(strTemp) > 0
            strTempBuffer = ssTemp.Value

            iCount = iCount + 1
            rss.ItemCount = iCount
            ' on rajoute un élément au tableau de résultats RSS
            ReDim Preserve rss.Items(iCount)
            ssTemp.Value = strTemp
            ' attribution des valeurs des tags au tableau RSS
            rss.Items(iCount).author = deHTMLentities(ssTemp.getNextXMLnode("author"))
            rss.Items(iCount).title = deHTMLentities(ssTemp.getNextXMLnode("title"))
            rss.Items(iCount).description = deHTMLentities(ssTemp.getNextXMLnode("description"))
            rss.Items(iCount).link = deHTMLentities(ssTemp.getNextXMLnode("link"))

            strItemize = strTempBuffer
```

```
            ssTemp.Value = strItemize
            strTemp = ssTemp.getNextXMLnode("item")

        Loop
    Loop
    Close #xmlFile
    Reset

    ' libération de la ressource
    Set ssTemp = Nothing

    t1 = GetTickCount()
    Debug.Print "parsing du xml : "; Format((t1 - t0) / 1000, "0.000") & " s"

    addRSStoTable "tblBlogs"

    t1 = GetTickCount()
    Debug.Print "avec table     : "; Format((t1 - t0) / 1000, "0.000") & " s"

End Function
```

D - Fonctions annexes

```
Public Function addRSStoTable(ByVal strTable As String)
    ' fonction qui permet de passer au Tableau d'éléments rss en mémoire
    '  à un stockage dans une table (tblBlogs)

Dim rec As DAO.Recordset
Dim i As Long

' ouverture d'un recordset sur la table tblBlogs
Set rec = CurrentDb.OpenRecordset("tblBlogs", dbOpenDynaset)

' nous parcourons le tableau rss qui est chargé en mémoire
For i = 1 To rss.ItemCount
    ' ajout d'un enregistrement
    rec.AddNew
    rec!date = Now
    rec!idTopic = i
    rec!Auteur = rss.Items(i).author
    rec!Titre = rss.Items(i).title
    rec!HyperLink = rss.Items(i).link
    rec!Message = deHTMLize(rss.Items(i).description)
    ' mise à jour du nouvel enregistrement
    rec.Update
Next i
rec.Close

' libération des ressources
Set rec = Nothing

End Function

Public Function deHTMLentities(ByVal str As String) As String
' fonction de nettoyage de chaîne
'  qui permet de rendre lisible des encodages destinés au HTML

Dim i As Integer

str = Replace(str, "&lt;", "<")
str = Replace(str, "&quot;", "'")
str = Replace(str, "&gt;", ">")
str = Replace(str, "&amp;", "&")
For i = 128 To 255
    str = Replace(str, "&#" & Format(i, "000") & ";", Chr(i))
Next i
str = Replace(str, "&amp;", "&")

deHTMLentities = str


End Function


Public Function deHTMLize(ByVal str As String) As String
' fonction destinée à nettoyer les balises HTML
```

```
'  pour rendre plus lisible un texte provenant d'un site

Dim i As Long
Dim strOut As String
Dim blnIntoTag As Boolean

str = Replace(str, "<br />", vbCrLf)

strOut = vbNullString
blnIntoTag = False
For i = 1 To Len(str)
    If Mid(str, i, 1) = "<" Then
        blnIntoTag = True
    ElseIf Mid(str, i, 1) = ">" Then
        blnIntoTag = False
    ElseIf Not blnIntoTag Then
        If Asc(Mid(str, i, 1)) = 10 Then
            'strOut = strOut & vbLf
        ElseIf Asc(Mid(str, i, 1)) = 9 Then
            'rien
        ElseIf Asc(Mid(str, i, 1)) = 13 Then
            strOut = strOut & vbCrLf
        ElseIf Asc(Mid(str, i, 1)) = 32 Then
            strOut = strOut & " "
        Else
            strOut = strOut & Mid(str, i, 1)
        End If
    End If
Next i

deHTMLize = strOut

End Function
```

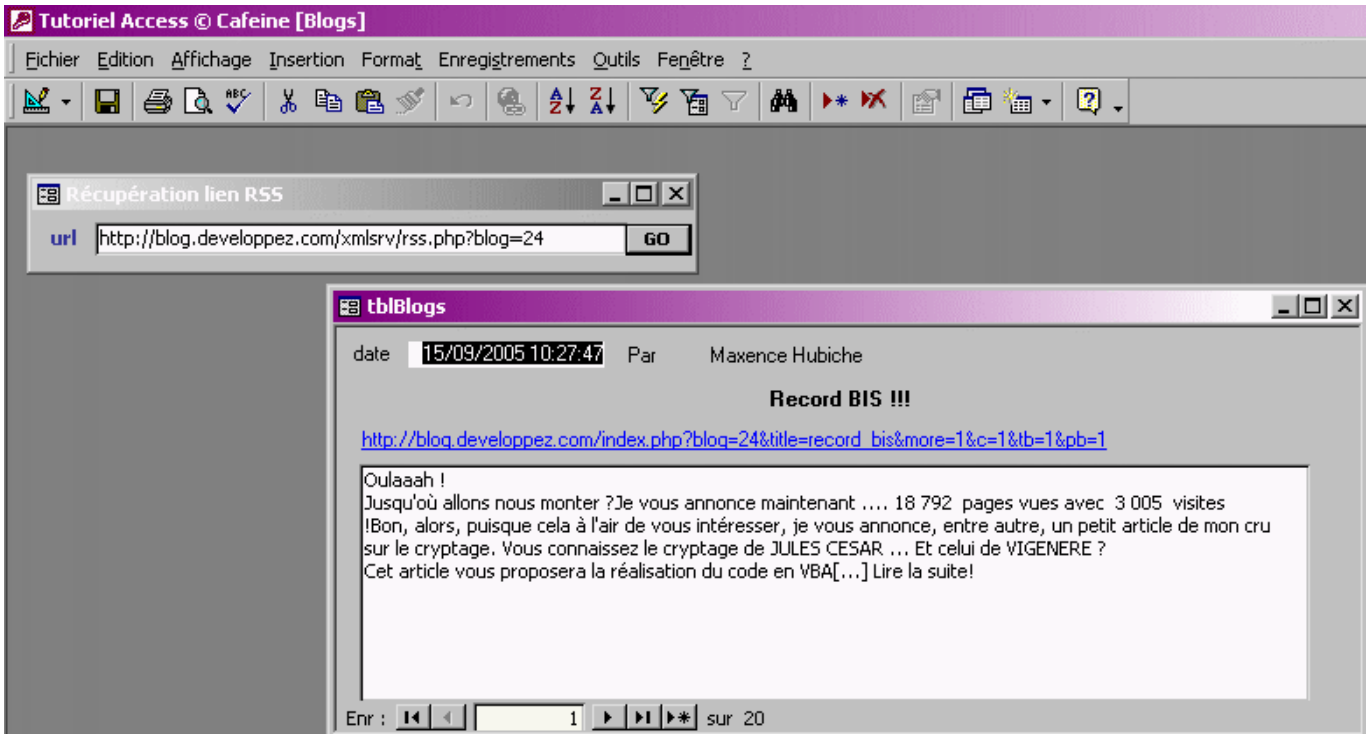
AddRSStoTable : permet d'insérer dans une table, l'intégralités des informations recueillies dans le tableau *rss*, qui est lui même issu du parsing.

Pour finaliser l'application, nous créons un formulaire.

Sur ce formulaire, nous créons une zone de texte (TextBox) qui contiendra l'url fichier à récupérer, et nous associons un bouton de commande qui va effectuer tout le process.

Et voilà, le tour est joué.

Pour rappel, voilà ce que cela donne :



V - Conclusions

Ce court tutoriel vise à ouvrir Access à des applications moins conventionnelles, ouvertes sur les technologies plus récentes (http, xml) et qui permettent de souligner la puissance de cet outil.

Vous pouvez télécharger la base de ce tutoriel en cliquant ici.

Merci de m'avoir lu ;).