Application des paramètres Batch : Temporisateur de base

par Jean-Philippe ANDRÉ (http://jpcheck.developpez.com)

Date de publication : 05 janvier 2010

Dernière mise à jour :

Cet article fait suite à la série d'articles sur les fichiers batch, leur contenu et Access. Pour compléter ce qui a pu être évoqué durant les différents tutoriaux existants, voici une base exemple qui permet de se rendre compte des possibilités qu'offrent les passages de paramètres.

I - Objectif du temporisateur....3
II - Notions de base déjà ouverte....3
III - Prise en compte du fichier ldb....3
IV - Code VBA....3
IV-a - Procédure Attendre....3
IV-b - Procédure Temporise....3
IV-c - Fonction Principale Startup....3
V - Code Batch....4
VI - Interface graphique....5
VII - Gestion avancée des paramètres....5
VII-a - Code VBA....5
VII-b - Contenu de la table....6
VII-c - Base à télécharger....7
VIII - Remerciements....7

I - Objectif du temporisateur

Pour éviter de chercher à ouvrir une base déjà utilisée par un autre utilisateur, l'applicatif proposé ici est de temporiser la demande du fichier batch et d'attendre que la base soit de nouveau disponible pour lancer les paramètres.

II - Notions de base déjà ouverte

Lorsqu'un fichier mdb est ouvert par un utilisateur, un fichier ldb est généré automatiquement dans le même répertoire. Pour indiquer à la machine qu'un utilisateur a potentiellement la main sur les tables que contient cette base. Pour éviter tout conflit d'accès aux données, on cherche donc à s'assurer que la base n'est pas déjà ouverte en vérifiant l'existence du fichier ldb en question. **Lien vers les fichiers ldb sur le site Microsoft**

III - Prise en compte du fichier ldb

En utilisant la fonction DoEvents, on va utiliser une fonction chronomètre qui va attendre un certain temps et tester à intervalle régulier l'existence du fichier, et continuer de tourner jusqu'à la disparition du fichier en question.

IV - Code VBA

IV-a - Procédure Attendre

Cette procédure temporise pendant le nombre de secondes qu'on lui transmet en argument :

```
Sub Attendre(Secondes As Integer)
' Cette procédure temporise pendant le nombre de secondes qu'on lui transmet en argument
Dim Début As Long, Fin As Long, Chrono As Long
Début = Timer
Fin = Début + Secondes
Do Until Timer >= Fin
    DoEvents
Loop
End Sub
```

IV-b - Procédure Temporise

On passe en paramètre à cette fonction le fichier qu'on souhaite voir " disparaître ", ainsi que le chemin de la base à lancer et enfin les paramètres à envoyer à cette base.

```
Sub Temporise(strpathfichier As String, strBasealancer As String, strparametres As String)
'Debug.Print Now()
Do Until Dir(strpathfichier) = ""
    Attendre (5)
Loop
Shell ("start /WAIT msaccess.exe """ & strBasealancer & """ ; """ & strparametres & """")
'Debug.Print Now()
End Sub
```

IV-c - Fonction Principale Startup

Comme dans les autres tutoriaux évoqués, on lance à l'ouverture de la base une macro AutoExec, qui exécute alors la fonction StartUp. Cette fonction récupère les paramètres passés (fichier à surveiller, base à lancer, paramètres à passer).

```
Public Function Startup()
Dim monparam    As Variant                                  ' déclare une variable
Dim RecuperationSplit As Variant
Dim RecuperationParam As Variant
Dim i As Integer
Dim j As Integer
    monparam = Command()                                ' affecte la valeur transmise à la variable

    If Not IsNull(monparam) Then    ' si la variable est nulle
        If Len(monparam) > 0 Then
                RecuperationSplit = Split(monparam, "|")

 Temporise CStr(RecuperationSplit(0)), CStr(RecuperationSplit(1)), CStr(RecuperationSplit(2))
                DoCmd.Quit acQuitSaveAll
        Else:
        End If
    End If
End Function
```

V - Code Batch

Dans un premier temps, on peut lancer la base en passant les paramètres sans vérifier que la base n'est pas déjà ouverte :

```
start /WAIT msaccess.exe "C:\temp\MaBdd.mdb" ; "MesParam"
```

Pour s'assurer que la base n'est pas déjà ouverte, on utilise le test d'existence en batch

```
if not exist "C:\temp\MaDbb.ldb"
```

On concatène les deux éléments dans la même instruction pour obtenir

```
if not exist "C:\temp\MaDbb.ldb" start /WAIT msaccess.exe "C:\temp\MaBdd.mdb" ; "MesParam"
```

Dans le second temps, lorsque le fichier ldb existe, on lance cette fois l'utilitaire pour qu'il temporise jusqu'à la disparition du fichier. La première est le test d'existence du fichier

```
if exist "C:\temp\MaDbb.ldb"
```

On lancera alors la base en lui fournissant les 3 éléments importants : le fichier à surveiller, le chemin de la base à lancer et les paramètres qu'on souhaite lui passer :

```
start /WAIT msaccess.exe "PathBaseTHOT" ; "Pathdufichier|Pathdelabase|Parametres"
```

ce qui nous donne en exemple :

```
start /WAIT msaccess.exe "C:\temp\THOT.mdb" ; "C:\temp\MaDbb.ldb|C:\temp\MaBdd.mdb|MesParam"
```

En concaténant les deux éléments dans la même instruction on arrive à

```
if exist "C:\temp\MaDbb.ldb" start /WAIT msaccess.exe "C:\temp\THOT.mdb" ; "C:\temp\MaDbb.ldb|C:\temp
\MaBdd.mdb|MesParam"
```

Et le fichier batch final qui permet de faire le lancement de la base de temporisation sera le suivant :

```
if not exist "C:\temp\MaDbb.ldb" start /WAIT msaccess.exe "C:\temp\MaBdd.mdb" ; "MesParam"
if exist "C:\temp\MaDbb.ldb" start /WAIT msaccess.exe "C:\temp\THOT.mdb" ; "C:\temp\MaDbb.ldb|C:\temp
\MaBdd.mdb|MesParam"
```

VI - Interface graphique

En attendant que le fichier soit lancé correctement, il est toujours plus agréable pour l'utilisateur qui doit savoir où en est le traitement de voir une interface qui lui indique la demande en cours.

Formulaire d'attente

VII - Gestion avancée des paramètres

L'idée est maintenant de passer toujours par l'**AutoExec**, mais de stocker les paramètres passés en dernier lieu par les différents batch. On va donc créer une table de paramètres qui permet de garder les informations nécessaires. Pour plus d'informations sur les tables de paramètres sous Access, je vous encourage à lire **cet article**.

VII-a - Code VBA

Au lieu de lancer directement la fonction **Temporise()** depuis la fonction **Startup()**, on récupère les données issues du batch et on ouvre le formulaire d'attente.

```
Public Function Startup()
Dim monparam    As Variant                                  ' déclare une variable
Dim RecuperationSplit As Variant
Dim RecuperationParam As Variant
Dim i As Integer
Dim j As Integer
    monparam = Command()                                ' affecte la valeur transmise à la variable

    If Not IsNull(monparam) Then    ' si la variable est nulle
        If Len(monparam) > 0 Then
                RecuperationSplit = Split(monparam, "|")
                SetGlobal "PathFichierCible", CStr(RecuperationSplit(0))
                SetGlobal "PathBaseaLancer", CStr(RecuperationSplit(1))
                SetGlobal "Paramètres", CStr(RecuperationSplit(2))
                DoCmd.OpenForm "Accueil"
                Form_Accueil.Accueil_BtnQuit_Click
                DoCmd.Quit acQuitSaveAll
```

```
        Else:
        'cas par défaut, rien :)
        End If
    End If

End Function
```

Et le code lié au formulaire sera le suivant :

```
Private Sub BtnQuit_Click()
    Temporise GetGlobal("PathFichierCible", "s"), GetGlobal("PathBaseaLancer", "s"),
 GetGlobal("Paramètres", "s")
    DoCmd.Close acForm, Me.Name
End Sub

Private Sub Form_Open(Cancel As Integer)
    Me.Label1.Caption = "En attente de la disparition du fichier" & vbCrLf &
 GetGlobal("PathFichierCible", "s")
End Sub


Sub Temporise(strpathfichier As String, strBasealancer As String, strparametres As String)
Dim strShellscript As String
Debug.Print Now()
Do Until Dir(strpathfichier) = ""
    Attendre (5)
Loop
strShellscript = "start /WAIT msaccess.exe """ & strBasealancer & """ ; """ & strparametres & """"
If Dir("C:\temp.bat") <> "" Then
    Kill "C:\temp.bat"
End If
EcrireFichierTexte "C:\temp.bat", strShellscript
ShellExecute Me.hwnd, "open", "C:\temp.bat", vbNull, vbNull, 5
End Sub


Public Sub Accueil_BtnQuit_Click()
    BtnQuit_Click
End Sub
```

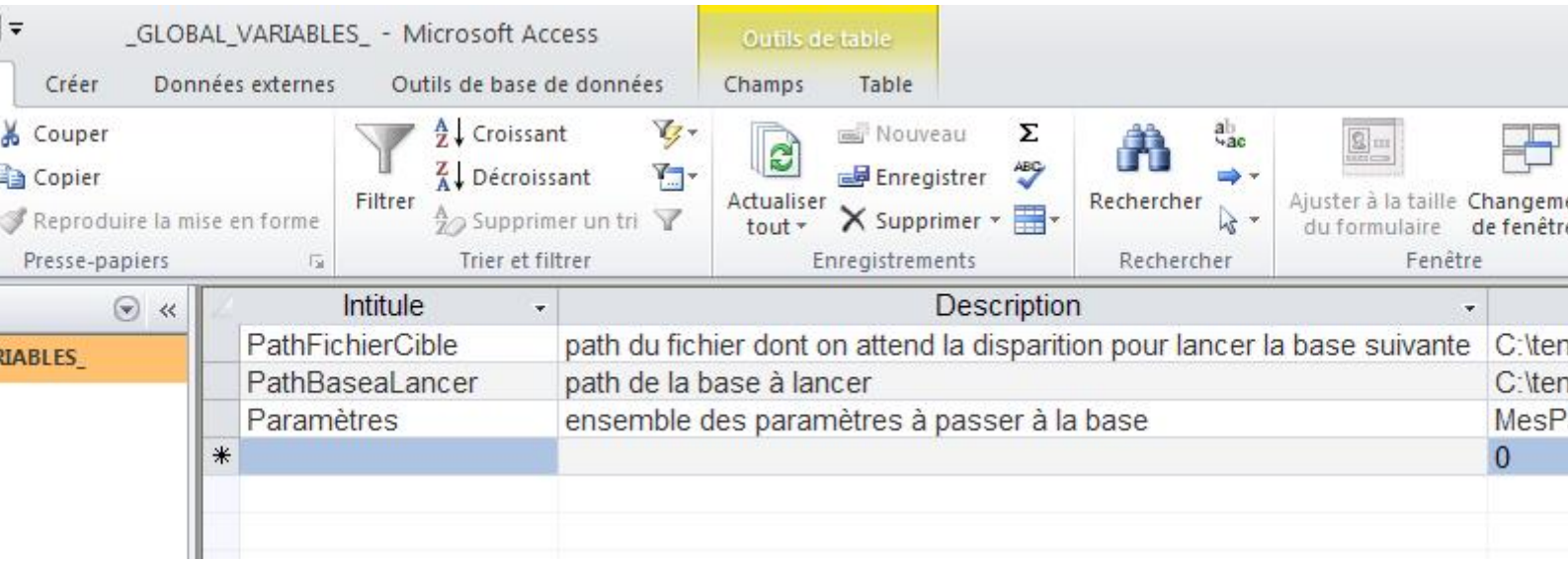
A noter qu'on utilise ici la fonction **ShellExecute()**

```
Public Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
(ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, _
ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

Cela permet notamment de ne plus lancer les batch directement en ligne de commande, mais d'ouvrir un fichier **.bat**

VII-b - Contenu de la table

Une fois le code exécuté, à l'ouverture du formulaire, voici le contenu de la table :



Contenu de la table de paramètres

VII-c - Base à télécharger

Lien

VIII - Remerciements

Je tiens à remercier l'équipe de Developpez.com pour la qualité du site, **AAA**, **BBB** et **CCC** pour la relecture de cet article, et de tous ceux qui contribuent à l'entraide autour du développement dans le cadre personnel et professionnel.