100% COURS
MCours.com
Cours et mémoires gratuits
mcourscom@gmail.com

Copyright © 2009 - Jean-Philippe ANDRÉ. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

I - Introduction

Cet article fait suite à la constatation du besoin de certains membres d'avoir une alternative aux variables globales, pour notamment des raisons de persistance en dehors de la durée de vie de l'application. Nous tacherons d'expliquer la pertinence de l'utilisation de tables de paramètres comme alternative à des stockages des données comme les fichiers. Elle permettra aux utilisateurs de gérer eux-mêmes des variables diverses et variées dans le cadre du développement de leurs applications. Cet article est plus spécialement destiné aux débutants, pour les familiariser avec les notions de paramétrage d'applications.

II - Avertissements

L'utilisation de la touche F1 est vivement conseillée à tous les stades de l'utilisation d'ACCESS. L'amélioration constante de l'aide en fait un partenaire de choix dans l'apprentissage permanent d'ACCESS. Personnellement je ne peux m'en passer, ne serait-ce que pour mémoire.

III - Pré-requis

Bien qu'elle ne soit pas obligatoire, je recommande la lecture de l'article sur **Les Fonctions de Domaine dans Access** de **Philippe JOCHMANS**. Les fonctions VBA évoquées sont documentées dans l'aide en ligne ainsi que dans la FAQ à votre disposition ici : **http://access.developpez.com/faq**.

IV - Table de paramètres

IV-A - Pourquoi ?

On peut utiliser des tables de paramètres pour de multiples raisons.

IV-A-1 - Alternative aux variables globales

Les variables globales n'ont de durée de vie que celle de l'ouverture de l'application. Ces variables sont déclarées dans un module, avec une portée Public. exemple

```
Public i As Integer
Public Const PathFichier = "C:\temp\fichiertemporaire.txt"
```

Le **cours d'initiation au VBA** d'**Olivier Lebeau** évoque très bien les durées de vie des variables en fonction de leur portée.
Lorsqu'on ouvre à nouveau le fichier, les variables sont réinitialisées et leurs dernières valeurs ne sont plus accessibles. Nous proposons ici de faire appel à des enregistrements dans des tables locales, qu'on appellera " table de paramètres ".

IV-A-2 - Alternative aux numéros auto-incrémentés

Il est courant d'utiliser des champs de type **NuméroAuto** dans les tables Access. Le champ s'incrémente tout seul de 1 à chaque nouvel enregistrement.
Le principal inconvénient est qu'en cas de suppression d'enregistrements, on se retrouve avec des "trous" dans la numérotation.

Copyright © 2009 - Jean-Philippe ANDRÉ. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

Exemple de table avec un trou de numérotation

Les tables de paramètres peuvent permettre de stocker le dernier numéro généré, ou encore plus pratique, indiquer le numéro du prochain enregistrement qui sera créé.

IV-A-3 - Alternative aux variables d'énumération

Les variables d'énumération sont des variables déclarées avec un type **Enum**. Variables et paramètres peuvent être déclarés avec un type Enum. Les éléments du type Enum sont initialisés à des valeurs constantes dans l'instruction Enum. Les valeurs affectées ne peuvent être modifiées au moment de l'exécution et peuvent comporter des nombres positifs et négatifs.
Exemple de variable Enum

```
Public Enum TaillePolice
 Petite = 8
 Standard = 12
 Grosse = 16
 Enorme = 18
End Enum
```

On peut donc avoir une énumération de valeurs, avec un intitulé qui s'y rattache.
Comme évoqué pour les variables globales, les variables d'énumération n'ont pas de valeurs dynamiques durant la durée de vie de l'application.

IV-B - Principe d'une table de paramètres

Le principe de stockage de données dans une table peut être basique (1 seul enregistrement pour 1 seule valeur dans une table) comme plus complexe (plusieurs champs, plusieurs variables). Dans le cas le plus souvent constaté, on a 2 ou 3 champs :

- intitulé de la variable
- valeur de la variable
- description de la variable (facultatif)

Tout comme on donne un nom à chaque variable dans le code VBA, on donne un intitulé à chaque variable/paramètre. Bien qu'on puisse stocker des chaînes de caractères avec des espaces, il est recommandé d'intituler ses variables sans ces espaces, mais en un seul mot, avec éventuellement des _ dans les intitulés.
Le deuxième champ important d'une table de paramètres est tout simplement le champ qui contient la valeur de la variable.
Ce champ peut-être de plusieurs types

- numérique
- date
- booléen

100% COURS
MCours.com
Cours et mémoires gratuits
mcourscom@gmail.com

Copyright © 2009 - Jean-Philippe ANDRÉ. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- texte

Pour des raisons de flexibilité, la solution utilisée dans les exemples sera celle utilisant un champ de type **texte**. Enfin, nous recommandons au moins un troisième champ, pour donner des détails sur l'utilisation de la variable. En effet, un intitulé de variable mal choisi, un nom pas adapté, et les autres développeurs qui auront à utiliser votre système auront toutes les peines du monde à savoir de quoi il s'agit si on ne les aide pas un peu.

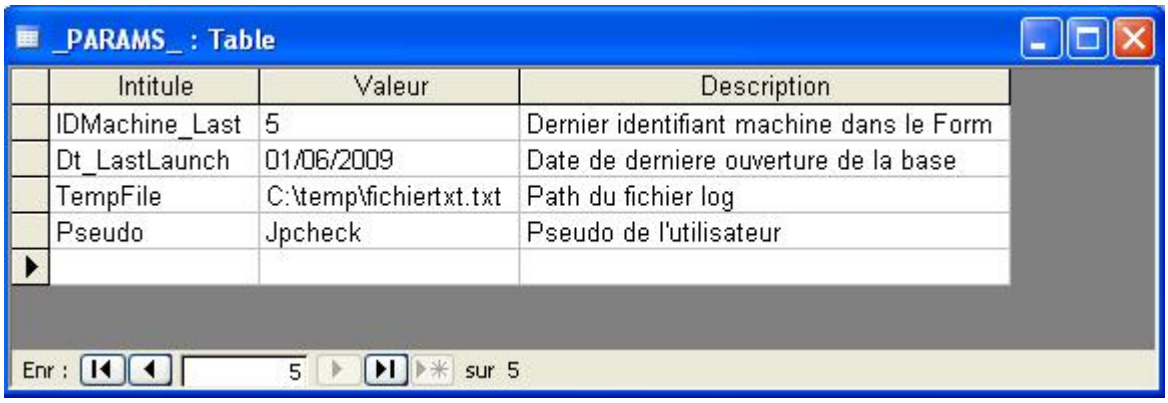
*Format de la table de paramètre _PARAMS_*

NB : dans tous les cas, l'intitulé doit être **unique**, aussi on le met en tant que clé primaire de notre table.

IV-C - Exemples de données stockées

On peut stocker n'importe quel type d'information qu'on souhaite voir perdurer dans le temps. De la même façon, des variables qui sont passées lors des appels de formulaires (paramètres **OpenArgs**). L'avantage de stocker un grand nombre de paramètres dans les tables permettra par la suite de simplifier le code.

Copyright © 2009 - Jean-Philippe ANDRÉ. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.



Exemples d'enregistrements dans une table de paramètres

On voit dans l'image ci-dessus qu'on peut aussi bien avoir des valeurs numériques (ici un identifiant machine), qu'une date ou encore un texte.
Les accès à cette donnée se feront aussi bien en lecture qu'en écriture pour qu'elle puisse évoluer en fonction des besoins de l'application.

NB : On donne dans l'exemple une date au format JJ/MM/AAAA, pour éviter les cas de versions anglophones, il est recommandé de garder une notation sous forme numérique (ici 39965 pour le 1er juin 2009).
Il en va de même pour les notations à partie décimale, avec les virgules **","** ou points **"."** selon les options de chaque machine.

IV-D - Code VBA

Il s'agit désormais de mettre en place les fonctions qui vont aller chercher ces variables, récupérer leur valeur, les mettre à jour. Il existe de nombreuses façons d'aller chercher la valeur d'un champ dans un enregistrement. Les méthodes de manipulations de données font l'objets de nombreux cours accessibles sur cette page : **Accès aux données**. Pour rendre le code le plus digeste et compact possible, on utilisera ici les fonctions de domaines.

IV-D-1 - Fonction de récupération de la valeur

La fonction VBA de base sera DLookup. La syntaxe de départ sera la suivante

```
Public Function GetParam(strWhat As String) As String
    GetParam = DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'")
End Function
```

Avec l'exemple des données dans la table "_PARAMS_"

```
MsgBox GetParam("Pseudo")
```

Affichera à l'écran

Copyright © 2009 - Jean-Philippe ANDRÉ. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.
http://jpcheck.developpez.com/tutoriels/access/table-parametres-initiation/

IV-D-2 - Fonction de mise à jour de la valeur

On utilisera ici une requête SQL. A moins de souhaiter une valeur en retour, on peut passer par une procédure. La syntaxe de départ sera la suivante

```
Public Sub SetGlobal(strWhat As String, strValue As Variant)
    CurrentDb.Execute "UPDATE _PARAMS_ SET valeur = '" & strValue & "' WHERE intitule='" &
 strWhat & "'"
End Sub
```

IV-D-3 - Utilisation dans les autres modules

On pourra utiliser les fonctions directement dans le reste du code

```
MsgBox GetParam("Pseudo")
SetGlobal "Pseudo","PiouPiou"
MsgBox GetParam("Pseudo")
```

IV-E - Valeurs par défaut ? Blindage des fonctions

Que faire si la variable n'existe pas ? Comment récupérer un autre type qu'une chaîne de caractères ? Quels sont les cas à anticiper ?
Dans un premier temps on s'assure qu'une valeur sera toujours retournée. On utilise la fonction **Nz()**

IV-E-1 - Valeur retour par défaut

```
Public Function GetParam(strWhat As String) As String
    GetParam = Nz(DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'"),"")
End Function
```

On retourne donc ici une chaîne vide si l'enregistrement retourne Null.

IV-E-2 - Typer le retour de la fonction

De plus, on peut décider de spécifier un type de retour avec la fonction GetParam. On spécifie en paramètre le type de données retour attendu. Le type de la fonction sera donc **Variant** et le code de la fonction peut se décomposer de la façon suivante :

```
Public Function GetParam(strWhat As String, strAs As String) As Variant
Select Case LCase(strAs)
    Case "int":
        GetParam = CInt(DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'"))
    Case "dt"
        GetParam = CDate(DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'"))
    Case "db"
        GetParam = CDbl(DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'"))
 Case Else
        GetParam = DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'")
End Select
End Function
```


Copyright © 2009 - Jean-Philippe ANDRÉ. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

IV-E-3 - Combinaison des deux

Enfin, puisqu'on souhaite s'assurer d'une valeur de retour dans tous les cas, on spécifie celle-ci après la fonction **Nz()** comme ceci :

```
Public Function GetParam(strWhat As String, strAs As String) As Variant
Select Case LCase(strAs)
    Case "int":
        GetParam = CInt(Nz(DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'"),0))
    Case "dt"
        GetParam = CDate(Nz(DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'"),#01/01/1950#))
    Case "db"
        GetParam = CDbl(Nz(DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'"),0))
 Case Else
        GetParam = Nz(DLookup("valeur", "_PARAMS_", "intitule='" & strWhat & "'"),"")
End Select
End Function
```

IV-E-4 - Blindage de la mise à jour

Certaines valeurs peuvent être problématiques, notamment les chaînes de caractères avec des apostrophes. On gère les caractères à risque en les doublant dans le code, comme suit :

```
Public Sub LetParam(strWhat As String, strValue As Variant)

 CurrentDb.Execute "UPDATE _PARAMS_ SET valeur = '" & Replace(strValue,"'","''") & "' WHERE intitule='" & Replace
End Sub
```

On s'autorise ainsi à stocker des variables avec des intitulés comme "Phrase d'accueil" et à stocker des valeurs elles aussi avec des apostrophes.

V - Plus loin avec les paramètres

En dehors de l'univers Access, vous avez la possibilité d'utiliser de nombreuses méthodes plus élaborées que le simple stockage dans une table. Les écueils que l'on rencontrera en utilisant des tables sont nombreux.:

- Univers multi-frontal : augmentation des champs de la table et obligation de garder une nomenclature efficace.
- Table locale vs table liée : les droits d'accès simultanés sont difficiles à appréhender sans étude préalable.
- Etc.

De même, les autres possibilités de pratiquer les paramètres dans les applications que l'on développe sont très nombreuses. Sans pouvoir donner une liste exhaustive, voici quelques pistes :

- Fichiers de registre (fichiers .reg)
- Fichiers ini : ces fichiers texte permettent de spécifier un ensemble de variables d'environnement d'une application
- Etc.

VI - Conclusion

Il est donc possible pour un utilisateur de gérer à sa propre manière les variables qui devront avoir une durée de vie plus importantes que celles "en dur" dans le code.
Quelques autres exemples d'utilisation de tables de paramètres

100% COURS
MCours.com
Cours et mémoires gratuits
mcourscom@gmail.com

Copyright © 2009 - Jean-Philippe ANDRÉ. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.

- personnalisation de l'interface en fonction de l'utilisateur (taille de police, boutons visibles ou pas, droits d'accès)
- utiliser l'application en multi-utilisateur sans avoir à manipuler des fichiers

VII - Remerciements

Je tiens à remercier l'équipe de Developpez.com pour la qualité du site, **Tofalu** et **Arkham46** pour la relecture de cet article, et de tous ceux qui contribuent à l'entraide autour du développement dans le cadre personnel et professionnel.

Copyright © 2009 - Jean-Philippe ANDRÉ. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérêts.