Programmation Objet : Classe String

par Charles A. [cafeine]

Date de publication : 20/10/2004

Dernière mise à jour : 20/10/2004

Créer et utiliser une classe dérivée du type String et lui ajouter des fonctionnalités

1- Définition d'une classe (cf. MSDN)

Petit rappel Théorique : **une classe**

Une classe est une définition d'objet. Elle contient des informations sur le comportement que devrait adopter l'objet, dont elle indique le nom, les méthodes, les propriétés et les événements. Elle n'est pas un objet en soi, en ce sens qu'elle n'existe pas en mémoire. Quand un code faisant référence à une classe est exécuté, une nouvelle instance de la classe, un objet, est créée en mémoire. Bien qu'il n'existe qu'une seule classe, plusieurs objets du même type peuvent être créés en mémoire à partir de cette classe.

Une classe peut être comparée à un objet « sur le papier » # autrement dit, elle fournit un schéma de l'objet, mais elle n'a aucune substance dans la mémoire. Il est possible de créer à partir de ce schéma un nombre illimité d'objets. Tous les objets créés à partir d'une classe sont dotés des mêmes membres (propriétés, méthodes et événements). Cependant, chaque objet se comporte comme une entité indépendante des autres. Par exemple, les propriétés d'un objet peuvent avoir des valeurs différentes de celles d'un autre objet du même type.

Un projet Microsoft® Visual Basic® pour Applications (VBA) peut contenir deux sortes de modules de classe : les modules de classe élémentaires, qui ne sont associés à aucune sorte d'interface utilisateur, et les modules de classe qui sont associés à un formulaire ou à un autre composant. Les modules de classe associés à un formulaire, par exemple, sont identiques aux modules de classes élémentaires, à cette exception près qu'ils ne sont présents en mémoire que tant que ce formulaire y est présent lui aussi. Les objets UserForms, les formulaires et les états Microsoft® Access, l'objet ThisDocument de Microsoft® Word, ainsi que les objets ThisWorkbook et SheetN de Microsoft® Excel sont tous des exemples d'objets associés à des modules de classe.

2- But de la classe du tutoriel : String Etendue

L'idée est de partir du type String, pour lui adjoindre des fonctions évoluées :

- . Calcul de la longueur de la chaîne
- . Mise en majuscules
- . Mise en minuscules
- . Calcul du nombre de mots
- . Récupération du Nième mot
- . Application d'une casse aléatoire
- . Susbtitution amusante de caractères
- . Mise en initiales
- . Position d'un mot
- . Remplacement d'une sous-chaîne (fonction existant dans Access2000 Replace())

3- Utilisation de la classe

Nous avons entièrement codé la classe, nous l'avons enregistrée sous le nom **clsString**

Nous pourrons à tout moment ajouter des fonctions ou des propriétés.

Nous pourrions tout à fait imaginer d'ajouter des fonctions supplémentaires comme par exemple du cryptage/décryptage.

Nous allons maintenant étudier comment utiliser cette classe.

```
Sub TestClasse()
Dim i As Integer
Dim sTest As New clsString

sTest.Value = "Salut ça va oui et toi ? bof pas mal mieux que toi ..."

Debug.Print "Valeur de la chaîne                              : " & sTest.Value
Debug.Print "casse aléatoire                                  : " & sTest.FunCase
Debug.Print "substitutions caract.                            : " & sTest.Lamer

Debug.Print "Longueur de chaîne                               : " & sTest.Length
Debug.Print "nombre de mots                                   : " & sTest.NbMots

Debug.Print "mise en initiales                                : " & sTest.Initiales
Debug.Print "position d'un mot 'toi' 2eme occurrence          : " & sTest.PositionMot("toi", 2)
Debug.Print "position d'un mot inexistant                     : " & sTest.PositionMot("Bizarre")
Debug.Print "remplacements ' ' par '%'                        : " & sTest.Replace(" ", "%")
Debug.Print "remplacements ' ' par '%' 2 fois                 : " & sTest.Replace(" ", "%", , 2)
Debug.Print "remplacements ' ' par '%' 2 fois, 2 occurrences  : " & sTest.Replace(" ", "%", 2, 2)
Debug.Print "occurrences 'toi'                                : " & sTest.NbOccurrences("toi")

End Sub
```

Nous mettons ce code dans un nouveau module, pour tester l'utilisation de cette classe.

Pour utiliser cette classe nous devons déclarer son type.

Dim NomDeLaVariable As New NomDeLaClasse

au lieu de

Dim NomDeLaVariable As String

Elle sera instanciée lors de la première attribution d'une propriété, dans notre cas : **.Value**

La sub exemple montre quelques utilisations possibles de cette classe.

Nous pouvons visualiser les résultats dans la fenêtre de débogage ou "immédiate".

```
TestClasse
Valeur de la chaîne                          : Salut ça va oui et toi ? bof pas mal mieux que toi
...
casse aléatoire                              : sALUT Ça VA ouI ET Toi ? boF paS mAl miEUX QuE TOI
...
substitutions caract.                        : §Ãlút çÃ V@ õúî &T tõí ? ßòf P@§ M@L mîÈúx qú& tõì
...
Longueur de chaîne                           : 54
nombre de mots                               : 14
mise en initiales                            : Salut Ça Va Oui Et Toi ? Bof Pas Mal Mieux Que Toi
...
position d'un mot 'toi' 2eme occurrence      : 13
position d'un mot inexistant                 : 0
remplacements ' ' par '%'                    :
Salut%ça%va%oui%et%toi%?%bof%pas%mal%mieux%que%toi%...
remplacements ' ' par '%' 2 fois             : Salut%ça%va oui et toi ? bof pas mal mieux que toi
...
remplacements ' ' par '%' 2 fois, 2 occurrences : Salut ça va%oui%et toi ? bof pas mal mieux que toi
...
occurrences 'toi'                            : 2
```

4- Code de la Classe

Ci-joint le code de la classe **clsString**

Aller dans les modules access, en créer un nouveau et l'enregistrer sous le nom **clsString**

```
Option Compare Database
Option Explicit

' Variable qui contient la chaîne
Private classStringValue As String

' Variable tableau qui contient les mots
Private classStringArray() As String

Private Sub Class_Initialize()
 classStringValue = vbNullString
End Sub
```

On définit la variable de type String : **classStringValue**

On définit la variable tableau de type String : **classStringArray()**

Ces variables vont stocker la valeur de la classe et les mots de la chaîne tout au long de l'existence de l'instance de cette classe, elles nous seront utiles pour effectuer des opérations sur la valeur de la classe.

Le portée de ces variables est **Private**, car elles ne seront utilisées qu'à l'intérieur de la classe.

```
Private Sub Class_Initialize()
 classStringValue = vbNullString
End Sub
```

La Sub **ClassInitialize()** permet d'initialiser la variable : classStringValue

```
Public Property Let Value(ByVal strData As String)
 classStringValue = strData

 'par défaut on remplit le tableau de mots avec le séparateur ESPACE
 'xSplit2() est la fonction auxiliaire qui découpe une chaîne en tableau
 ' dans le tableau classStringArray
 ' cette fonction sera décrite un peu plus loin dans le tutoriel
 xSplit2 classStringValue
End Property
```

Property Let : indique que la propriété en question va recevoir une valeur, la valeur en question est l'argument de la Propriété (dans notre cas : strData).

Ici on attribue la valeur de vData à notre variable Private **classStringValue**.

Public : car cette propriété doit être accessible depuis l'extérieur de la classe.

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

Property Get : indique que la valeur de la propriété en question va être renvoyée à l'utilisateur.

Comme dans une fonction VBA, "NomDeLaFonction = " renvoie la valeur. Cette propriété renverra la valeur de notre variable Private **classStringValue**.

Abordons maintenant nos premières fonctions simples :

```
Public Function Length() As Long
 Length = Len(classStringValue)
End Function

Public Function Maj() As String
 Maj = UCase(classStringValue)
End Function

Public Function Min() As String
 Min = LCase(classStringValue)
End Function
```

Ces fonctions de classes s'écrivent comme des fonctions VBA classiques. Le type est à nouveau **Public** pour que ces fonctions soient accessibles depuis l'extérieur du module de classe.

Cette fonction a pour but de calculer la longueur de la chaîne, la fonction VBA requise est *Len()*.

La fonction manipule notre variable Private **classStringValue** et en renvoie la longueur.

Les deux fonctions suivantes utilisent *LCase()* pour la mise en minuscules et *UCase()* pour la mise en majuscules.

5- Fonctions de classe avancées

```
Public Property Get NbMots(Optional strDelim As String = " ") As Integer

 ' découpe la chaîne en fonction du délimiteur
  xSplit2 classStringValue, strDelim

 ' renvoie la taille du tableau de mots
 '  UBound(VariableDeTypeTableau) renvoie la taille du tableau
 '  Nous utilserons souvent cette fonction pour parcourir notre tableau classStringArray
 NbMots = UBound(classStringArray)

End Property
```

Renvoie le nombre de mots de la chaîne en fonction d'un délimiteur choisi par l'utilisateur.

Nous utilisons un argument facultatif avec le mot Clé **Optional**. On attribue une valeur par défaut, le caractère ESPACE en mettant = " ".

```
Public Property Get Mot(ByVal intIndex As Integer, Optional ByVal strDelim As String = " ")

' découpe la chaîne en fonction du délimiteur
xSplit2 classStringValue, strDelim
If intIndex <= UBound(classStringArray) Then
    ' si le numéro du mot recherché est inférieur à la taille du tableau, on renvoie le mot
    Mot = classStringArray(intIndex)
Else
    ' sinon on renvoie une chaîne nulle
    Mot = vbNullString
End If

End Property
```

Renvoie le mot correspondant au numéro demandé par l'utilisateur.

```
Public Function FunCase() As String
 Dim out As String
 Dim i As Integer
 Dim hasard As Integer

 Randomize 'initialisation des nombres aléatoires
 out = ""
 If Len(classStringValue) > 0 Then
    For i = 1 To Len(classStringValue) 'parcourt la chaîne caractère par caractère
        hasard = Int((Rnd * 10) + 1) 'calcule une valeur entre 1 et 10
        ' si la valeur est > 5 la lettre sera en majuscule, sinon elle sera en minuscule
        out = out & IIf(hasard > 5, UCase(Mid(classStringValue, i, 1)), LCase(Mid(classStringValue,
i, 1)))
    Next i
    FunCase = out
 Else
    FunCase = vbNullString
 End If
End Function
```

Renvoie une chaîne avec une casse (majuscule/minuscule) aléatoire.

```
Public Function Lamer() As String
Dim out As String
Dim i, pos As Integer
Dim samlam() As String
Dim sample As String
Dim hasard As Integer

ReDim samlam(4)
'chaîne qui contient les caractères remplaçables
```

```
    sample = "abcdefghijklmnopqrstuvwxyz1234567890"
'Tableau des chaînes qui contiennent les caractères spéciaux
samlam(1) = "@ß©þêFGHíjKLmÑøP¶®§TúvWX¥z1²³4SG789o"
samlam(2) = "äB¢DÉFghîJk|MñõpqrßtûVwxµZ¹²345G7890"
samlam(3) = "âB©ÐÉfGHîjKLMÑòpqR$TúvwXÝz¹²³4S6789o"
samlam(4) = "Ãß©þ&fghìJklmñõP¶®§tüVWx¥Z¹2³4567890"

Randomize 'on initialise le tirage aléatoire
out = ""
If Len(classStringValue) > 0 Then
    For i = 1 To Len(classStringValue) 'on parcourt la chaîne avec une boucle
        If InStr(sample, Mid(classStringValue, i, 1)) Then
            ' si le caractère est remplaçable ...
            pos = InStr(sample, Mid(classStringValue, i, 1)) 'on détermine la position de la lettre
à remplacer
            hasard = Int(Rnd * 4) + 1 'on choisit un chiffre au hasard entre 1 et 4
            out = out & Mid(samlam(hasard), pos, 1) 'on écrit le caractère de substitution ...
        Else
            ' le caractère n'est pas remplaçable on laisse celui d'origine
            out = out & Mid(classStringValue, i, 1)
        End If
    Next i
    Lamer = out
Else
    Lamer = vbNullString
End If

End Function
```

Renvoie une chaîne avec des caractères substitués et choisis aléatoirement parmi 4 listes.

```
Public Property Get Replace(ByVal strSearch As String, _
                            ByVal strReplac As String, _
                            Optional ByVal IntStart As Integer = 0, _
                            Optional ByVal intCount As Integer = 2 ^ 16 / 2 - 1) As String
' ------------------------------------------------
' strSearch     :   chaîne recherchée
' strReplac     :   chaîne qui se substituera
' IntStart      :   Nombre d'occurrences à partir de laquelle commenceront les remplacements (par
défaut 0)
' IntCount      :   Nombre de valeurs qui seront remplacées (par défaut le maximum pour un type
Integer)
' ------------------------------------------------

Dim i, iCount, iStart As Integer
Dim strOut As String 'chaîne qui contiendra le résultat
iCount = 0
iStart = 0

If Len(classStringValue) > 0 Then
 i = 1
 ' Boucle qui parcourt la chaîne
 Do While i <= Len(classStringValue)
  If Mid(classStringValue, i, Len(strSearch)) = strSearch And iCount < intCount Then
   'cas d'une occurence trouvée et le nb maximum de remplacements n'est pas atteint
   If iStart >= IntStart Then
    strOut = strOut & strReplac
    i = i + Len(strSearch)
    iCount = iCount + 1 'on incrémente le compteur de remplacements
    iStart = iStart + 1 'on incrémente le compteur d'occurences trouvées
   Else 'le nombre de remplacement est atteint on laisse remet le caractère d'origine
    iStart = iStart + 1
    strOut = strOut & Mid(classStringValue, i, 1)
    i = i + 1
   End If
  Else  'il n'y a pas de remplacement à faire la chaîne recherchée n'est pas trouvée
   strOut = strOut & Mid(classStringValue, i, 1)
   i = i + 1
  End If
 Loop

 Replace = strOut
Else
 Replace = ""
End If

End Property
```

Renvoie la chaîne d'origine avec les remplacements demandés.

Cette fonction est apparue de manière native à partir des versions 2000 d'Access et ultérieures.

```
Public Property Get Initiales(Optional strDelim As String = " ") As String
Dim i As Integer

'découpage de la chaîne en fonction du délimiteur
xSplit2 classStringValue, strDelim

For i = 1 To UBound(classStringArray) 'parcourt le tableau de mots
    'met en majucule le premier caractère du mot
    classStringArray(i) = UCase(Left(classStringArray(i), 1)) & LCase(Right(classStringArray(i),
Len(classStringArray(i)) - 1))
Next i

'recombination de la chaîne à partir des mots mis en majuscules et avec le délimiteur
Initiales = xJoin2(strDelim)

End Property
```

Renvoie la chaîne d'origine avec la première lettre de chaque mot en majuscules.

```
Public Property Get NbOccurrences(ByVal strMot As String, Optional strDelim As String = "") As
Integer
' ------------------------------------------------
' Mot           :   Mot recherché
' intCount      :   Nombre d'occurrences du mot
' strDelim      :   chaîne qui délimite les mots
' ------------------------------------------------
Dim i As Integer

'découpage de la chaîne en fonction du séparateur
xSplit classStringValue, strDelim

'compteur d'occurrences du mot
NbOccurrences = 0
For i = 1 To UBound(classStringArray) 'on parcourt le tableau des mots
    If classStringArray(i) = strMot Then
    ' si le mot correspond
      NbOccurrences = NbOccurrences + 1
    End If
Next i

End Property
```

Renvoie le nombre d'occurrences d'un mot dans la chaîne.

```
Public Property Get PositionMot(ByVal strMot As String, Optional intCount As Integer = 1, Optional
strDelim As String = "") As Integer
' ------------------------------------------------
' Mot           :   Mot recherché
' intCount      :   Nombre d'occurrences du mot à partir de laquelle on donne la position
' strDelim      :   chaîne qui délimite les mots
' ------------------------------------------------
Dim i, xCount As Integer

'découpage de la chaîne en fonction du séparateur
xSplit classStringValue, strDelim

'compteur d'occurrences du mot
xCount = 1
For i = 1 To UBound(classStringArray) 'on parcourt le tableau des mots
    If classStringArray(i) = strMot Then
    ' si le mot correspond
        If xCount = intCount Then
        ' si l'occurence correspond on renvoie la position du mot
            PositionMot = i
            Exit Property
        Else
```

```
            ' sinon on incrémente le compteur d'occurrences
                xCount = xCount + 1
            End If
        End If
    Next i
    PositionMot = 0

    End Property
```

Renvoie la position du mot recherché dans la chaîne après la Nième occurrence.

6- Fonctions auxiliaires spécifiques à Access97

Nous avons besoin de découper notre chaîne de manière simple et rapide.

Depuis Access2000, nous disposons de *Split()* et *Join()*, mais ces fonctions ne sont pas hélas dans Access97.

Nous allons pallier à ces lacunes en créant des équivalents.

Ces fonctions vont s'intégrer à la classe pour un usage interne et seront donc définies en tant que **Private**.

Ces fonctions utiliseront les deux variables de portée locale **classStringValue** et **classStringArray**.

```
Private Function xSplit2(ByVal strExpr As String, Optional ByVal strDelim As String = " ") As
Boolean
Dim i, Count As Integer
Dim strOut As String

strOut = vbNullString

' initialise notre tableau à un élément
ReDim classStringArray(1)
classStringArray(1) = vbNullString

strExpr = strExpr & strDelim 'on rajoute un délimiteur de plus pour écrire le dernier mot dans le
tableau
For i = 1 To Len(strExpr) 'parcourt la chaîne de caractère lettre par lettre
 If Mid(strExpr, i, Len(strDelim)) = strDelim Then
  'si le séparateur de mot est trouvé
  '  on incrémente le compteur on écrit le mot dans le tableau
  '  grâce à la Sub Mots()
  Count = Count + 1
  Mots Count, strOut
  strOut = vbNullString
 Else
  ' si le séparateur n'est pas trouvé
  '  on ajoute la lettre au mot en cours
  strOut = strOut & Mid(strExpr, i, 1)
 End If
Next i

End Function

Private Function xJoin2(Optional ByVal strDelim As String = " ") As String
Dim i, Count As Integer

'on parcourt notre tableau de mots
For i = 1 To UBound(classStringArray)
 ' on ajoute chaque mot suivi du délimiteur à la chaîne résultat
 xJoin2 = xJoin2 & classStringArray(i) & strDelim
Next i
' on enlève le dernier délimiteur
xJoin2 = Left(xJoin2, Len(xJoin2) - Len(strDelim))
End Function


Private Sub Mots(ByVal intIndex As Integer, ByVal strExpr As String)

' si l'index est supérieur à la taille du tableau
'  on redimensionne le tableau en conservant les anciennes valeurs (Preserve)
If UBound(classStringArray) < intIndex Then
 ReDim Preserve classStringArray(intIndex)
End If

' affectation de du mot dans le tableau
```

```
    classStringArray(intIndex) = strExpr
End Sub
```

7- Fonctions de Chaînes de Caractères

Dans ce tutoriel nous avons abordé parmi les plus courantes fonctions de chaînes.

Nous les étudierons dans ce tutoriel dans leur utilisation la plus simple, sans comparaison de casse.

Len(chaîneDeCaractères)

Renvoie la longueur d'une chaîne en type Long

```
print Len("ABCDEFGHIJ")
 10

print Len("")
 0
```

Mid(chaîneDeCaractères, PositionDébut, *Longueur*)

Renvoie une sous-chaîne de caractères

```
print Mid("ABCDEFGHIJ",2,2)
BC

print mid("ABCDEFGHIJ",2)
BCDEFGHIJ

print mid("ABCDEFGHIJ",2,50)
BCDEFGHIJ
```

InStr(chaîneDeCaractères, chaîneRecherchée)

Renvoie la position d'un chaîne de caractère à l'intérieur d'une autre qui est censée la contenir.

```
print InStr("ABCDEFGHIJ","EFG")
 5

print InStr("ABCDEFGHIJ","X")
 0
```

Right(chaîneDeCaractères, NombreDeCaractères)/Left(chaîneDeCaractères, NombreDeCaractères)

Renvoie les n caractères d'une chaîne en partant de la droite ou de la gauche.

```
print Left("ABCDEFGHIJ", 3)
ABC

print Left("ABCDEFGHIJ", 255)
ABCDEFGHIJ

print Right("ABCDEFGHIJ", 5)
FGHIJ
```

LCase(chaîneDeCaractères)/UCase(chaîneDeCaractères) Modifie la casse d'une chaîne de caractères :

LCase() (LowerCase) met en minuscules, et **UCase()** (UpperCase) met en majuscules.

```
print LCase("ABCDEFGHIJ1234567890")
abcdefghij1234567890

print UCase("abcdefghijéêèàâîôùû")
ABCDEFGHIJÉÊÈÀÂÎÔÙÛ
```

8- Conclusion

Ce tutoriel très simple vous aura, je l'espère, familiarisé avec le concept et l'utilisation de classes, voire vous aura donné des idées pour vos futurs développements.

Vous pouvez télécharger ce tutoriel au format zip [3 ko] en cliquant ici.