

INITIATION

AU

LOTUS SCRIPT

TABLE DES MATIERES

INTRODUCTION.....	4
CONNAISSANCE PRE - REQUISE.....	5
LES DIFFERENTS OBJETS TRAITES DANS CE TUTORIAL.....	5
1 L'AIDE EN LIGNE DU DESIGNER.....	6
2 OÙ PUIS-JE UTILISER DU LOTUS SCRIPT ?.....	8
3 PRINCIPE DE BASE : LA DIFFERENCE ENTRE FRONTAL ET DORSAL.....	10
3.1 ARBORESCENCE DES PRINCIPAUX OBJETS.....	10
3.1.1 Classes frontales.....	10
3.1.2 Classes dorsales.....	11
4 LES VARIABLES.....	12
4.1 CONVERSION DES VARIABLES.....	13
4.2 LA VARIABLE TABLEAU.....	13
4.2.1 Déclaration d'un tableau.....	13
4.2.2 Manipulation d'un tableau.....	14
4.2.3 Connaître la taille d'un tableau.....	15
4.2.4 Utilisation de « Option Base ».....	16
4.3 LA VARIABLE LISTE.....	16
4.3.1 Déclaration d'une liste.....	16
4.3.2 Manipulation d'une liste.....	17
4.4 UN PETIT TRUC A PROPOS DU NOMMAGE DES VARIABLES.....	18
5 ACCEDER A LA BASE NOTES EN COURS.....	20
5.1 DECLARATION ET OUVERTURE D'UNE SESSION.....	20
5.2 DECLARATION ET OUVERTURE DE LA BASE LOTUS NOTES EN COURS.....	20
6 LA GESTION DES DOCUMENTS.....	21
6.1 LA DIFFERENCE ENTRE UN DOCUMENT ET UN MASQUE.....	21
6.2 DECLARATION ET CREATION D'UN DOCUMENT.....	21
6.3 MANIPULER LES CHAMPS D'UN DOCUMENT.....	21
6.4 LE MASQUE AFFECTE AU DOCUMENT.....	22
6.5 LES COMMANDES DE BASES D'UN DOCUMENT.....	22
6.6 IDENTIFIANT UNIQUE & IDENTIFIANT UNIVERSEL.....	23
6.7 LA GESTION DES CHAMPS DANS UN DOCUMENT.....	24
6.7.1 L'objet « NotesItem ».....	25
6.8 LA GESTION DES CHAMPS DE TYPE « TEXT RICHE ».....	27
6.8.1 Déclaration et initialisation.....	27
6.8.2 Manipulation d'un champ « Rich Text ».....	28
6.8.3 Insérer un « LienDoc » dans un champ « Rich Text ».....	28
7 LES VUES & LES COLLECTIONS.....	29
7.1 LES VUES.....	29
7.1.1 Initialiser un objet NotesView.....	29
7.1.2 Manipulation d'une vue.....	29
7.1.3 Rechercher un document dans une vue en fonction d'un mot clé.....	30
7.2 LES COLLECTIONS DE DOCUMENTS.....	31
7.2.1 Initialisation d'une collection.....	31
7.2.2 Manipulation d'une collection de documents.....	31
7.3 DIFFERENCE ENTRE UN « SEARCH » ET UN « GETALLDOCUMENTBYKEY ».....	33
8 GESTION DES NOMS NOTES.....	34
8.1 INITIALISER UN OBJET NOTESNAME.....	34
8.2 MANIPULER L'OBJET NOTESNAME.....	34
8.3 A PROPOS DES CHAMPS DE TYPE NOMS.....	35
9 LES AGENTS.....	36
9.1 INITIALISATION DE L'AGENT.....	36
9.2 LANCEMENT DE L'AGENT.....	36
9.3 PASSER UN DOCUMENT EN PARAMETRE.....	36

10 PROGRAMMATION FRONTALE.....	39
10.1 L'OBJET NOTESUIWORKSPACE.....	39
10.1.1 Initialiser l'objet NotesUIWorkSpace	39
10.1.2 Manipuler l'objet NotesUIWorkSpace	39
10.1.3 LES BOITES DE DIALOGUE.....	40
10.2 L'OBJET NOTESUIDOCUMENT.....	40
11 LES BIBLIOTHEQUES DE SCRIPTS.....	43
11.1 CREER UNE FONCTION.....	43
11.2 CREER UNE PROCEDURE	44
11.3 OPTION PUBLIC	46
11.4 UTILISATION D'UNE BIBLIOTHEQUE DE SRIPT	47
11.5 LE CHARGEMENT D'UN BIBLIOTHEQUE DE SCRIPT.....	47
12 DES PRECISIONS SUR DIVERS ASPECTS DU LOTUS SCRIPT	48
12.1 DECLARATION RAPIDE D'UN OBJET	48
12.2 FRACTIONNER UNE INSTRUCTION SUR PLUSIEURS LIGNES	48
12.3 LA COMMANDE « FORALL »	49
12.4 GESTION DES ERREURS	49
12.5 UTILISER LES FORMULES EN LOTUS SCRIPT	51
12.5.1 Restrictions.....	51
12.5.2 Utilisation.....	51
12.6 PASSER UNE CHAINE DE CARACTERES	52
12.7 LES COMMENTAIRES	52
13 L'UTILISATION DU « DEBOGUEUR LOTUS SCRIPT »	54
<u>CONCLUSION (IL EN FAUT BIEN UNE)</u>	<u>57</u>

Tous le contenu de cette création est mise à disposition sous un [contrat Creative Commons](#).



Cette création est mise à disposition selon le Contrat Paternité

- Pas d'Utilisation Commerciale
- Pas de Modification
-

disponible en ligne <http://creativecommons.org/licenses/by-nc-nd/2.0/fr/>

ou par courrier postal à :

Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

michael.delique@gmail.com

INTRODUCTION

Le Lotus Script (ou LS pour les initiés) est le langage le plus utilisé dans Lotus Notes (d'autres langages sont supportés : Java, Java Script...). Bien qu'il soit à première vue plus complexe à maîtriser que le langage de formules (ou @Formules), il est beaucoup plus souple à utiliser et offre bien plus de possibilités. Cela ne veut pas dire que vous devez renoncer au langage de formules, ce dernier est parfois plus pratique et puissant que le Lotus Script (il permet de faire en une ligne ce qui en nécessite plusieurs en Lotus Script). L'utilisation de l'un ou de l'autre dépendra de votre appréciation.

En règle générale, lorsque votre programme en formules comporte des boucles (à partir de la Version 6 de Lotus Notes le langage de formule permet de faire des boucles) ou devient complexe, l'on préfère passer au Lotus Script, cela facilite grandement la maintenance.

Ce tutorial est une initiation, il vous expliquera les grandes lignes afin que vous puissiez débiter en Lotus Script. Je vous recommande vivement d'aller acheter un livre sur le sujet (il en existe plusieurs très bien faits), et en tout état de cause cela ne remplacera JAMAIS une formation en bonne et due forme. Lotus Notes n'est pas un logiciel de bureautique comme Word ou Excel mais un système client-serveur complexe et je dirais même subtile.

En fonction de la version de Lotus Notes que vous utilisez (4.6, 5, 6, 6.5 ou 7) il y a quelques variantes. Mais dans l'essentiel c'est la même chose.

Ce tutorial ne vous donnera pas dans le détail l'explication de toutes les propriétés et méthodes liées aux objets. Vous disposez de l'aide en ligne du designer qui vous donnera toutes les informations sur les paramètres et conditions d'utilisations, elle est votre meilleur alliée.

Les codes présentés sont des exemples, le mieux est de faire des tests, par sois même, en même temps pour comprendre comment ça marche.

Pour information : toutes les captures d'écrans présentées sont réalisées à partir de la version 6.5 de Lotus notes.

CONNAISSANCE PRE - REQUISE

Le présent tutorial n'a pas pour but de vous initier aux joies de l'algorithmique et de la programmation en général, mais de vous donner les bases du Lotus Script. Vous devez donc savoir ce qu'est une variable (et son type), les expressions conditionnelles (if), les boucles (while, do, for)...

Si vous n'avez pas ces connaissances vous trouverez sur Internet de nombreux sites ou dans le commerce d'excellents livres qui vous donneront les bases nécessaires.

Vous devez aussi avoir une petite connaissance de Lotus Notes, savoir ce qu'est un masque, une vue et aussi savoir les créer via le designer. En fait si vous lisez ce tutorial, c'est que vous faites déjà du développement avec Lotus Notes et que vous avez besoin de passer au Lotus Script car vous avez épuisé les possibilités qu'offrent les @Formules.

LES DIFFERENTS OBJETS TRAITES DANS CE TUTORIAL

- NotesSession
- NotesDatabase
- NotesDocument
- NotesItem
- NotesView
- NotesDocumentCollection
- NotesName
- NotesRichTextItem
- NotesAgent
- NotesUlworkSpace
- NotesUIDocument

1 L'AIDE EN LIGNE DU DESIGNER

Il existe trois aides en ligne différentes, une par client (Notes, Designer et Administrator). Pour accéder à l'aide en ligne du designer, ouvrez le Designer et appuyez sur la touche « F1 », l'aide en ligne s'ouvrira et une icône apparaîtra sur l'espace de travail de votre client Lotus Notes. L'aide en ligne est en fait une base Lotus Notes comme les autres.

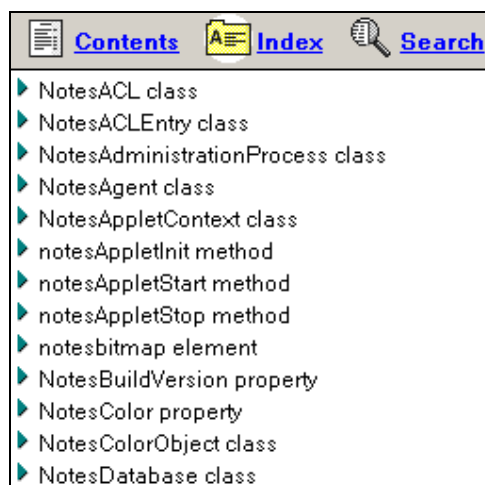
L'aide en ligne regroupe toutes les informations nécessaire au développement. Tous les principes de bases sont expliqués. Toutes les @formules, commandes et objets (Lotus Script ou Java) sont détaillés, leurs paramètres expliqués et leurs limites indiquées, de nombreux exemples d'utilisations sont aussi fournis.

L'aide en ligne propose trois méthodes de recherche de l'information :

- Via une arborescence (thèmes, chapitres, sections...).

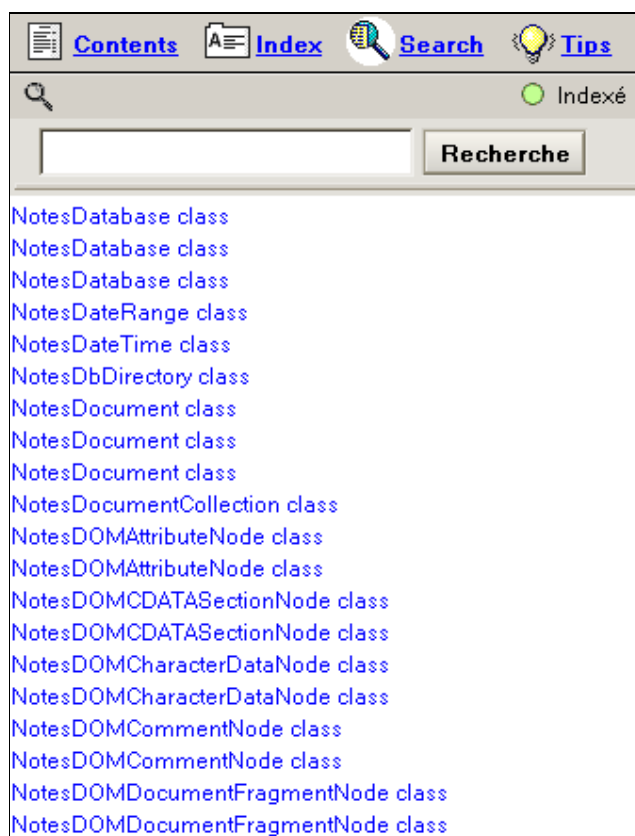


- Via une liste de mots triés par ordre alphabétique.



INITIATION AU LOTUS SCRIPT

- Via une recherche « full-text ».



Vous trouverez la réponse à de nombreuses questions grâce à l'aide en ligne. Elle est un de vos outils de développement privilégiés. Aucun développeur ne peut travailler sans, avec un peu de pratique vous trouverez facilement l'information que vous désirez.

Au fur et à mesure que vous lirez ce tutorial, consultez l'aide en ligne pour apprendre à l'utiliser et obtenir plus de détail sur les objets, méthodes et propriétés cités.

L'aide en ligne est une véritable mine d'informations qui bien exploitée peut vous faire gagner beaucoup de temps et d'énergie. Elle peut aussi, dans une certaine mesure, vous aider à apprendre par vous-même grâce aux nombreux exemples fournis. Elle est bien documentée, n'hésitez pas à l'utiliser.

Malheureusement, l'aide en ligne est en anglais, même lorsque vous avez une version française une partie reste en anglais. Une bonne pratique de l'anglais n'est, heureusement, pas nécessaire pour pouvoir la comprendre et Internet fournit de nombreux sites gratuits de traduction automatique. Il ne faut pas que vous vous amputiez d'une telle source d'information rapidement disponible (car elle est en local sur votre disque dur) parce que la première approche est un peu difficile du fait de la langue ou que vous vous sentez perdu sous l'avalanche d'informations. Vous apprendrez très vite à vous y retrouver.

Pour résumer en quelques mots :

UTILISEZ L'AIDE EN LIGNE, ELLE A UN CONTENU TRES RICHE ET VOUS AVEZ TOUT A Y GAGNER

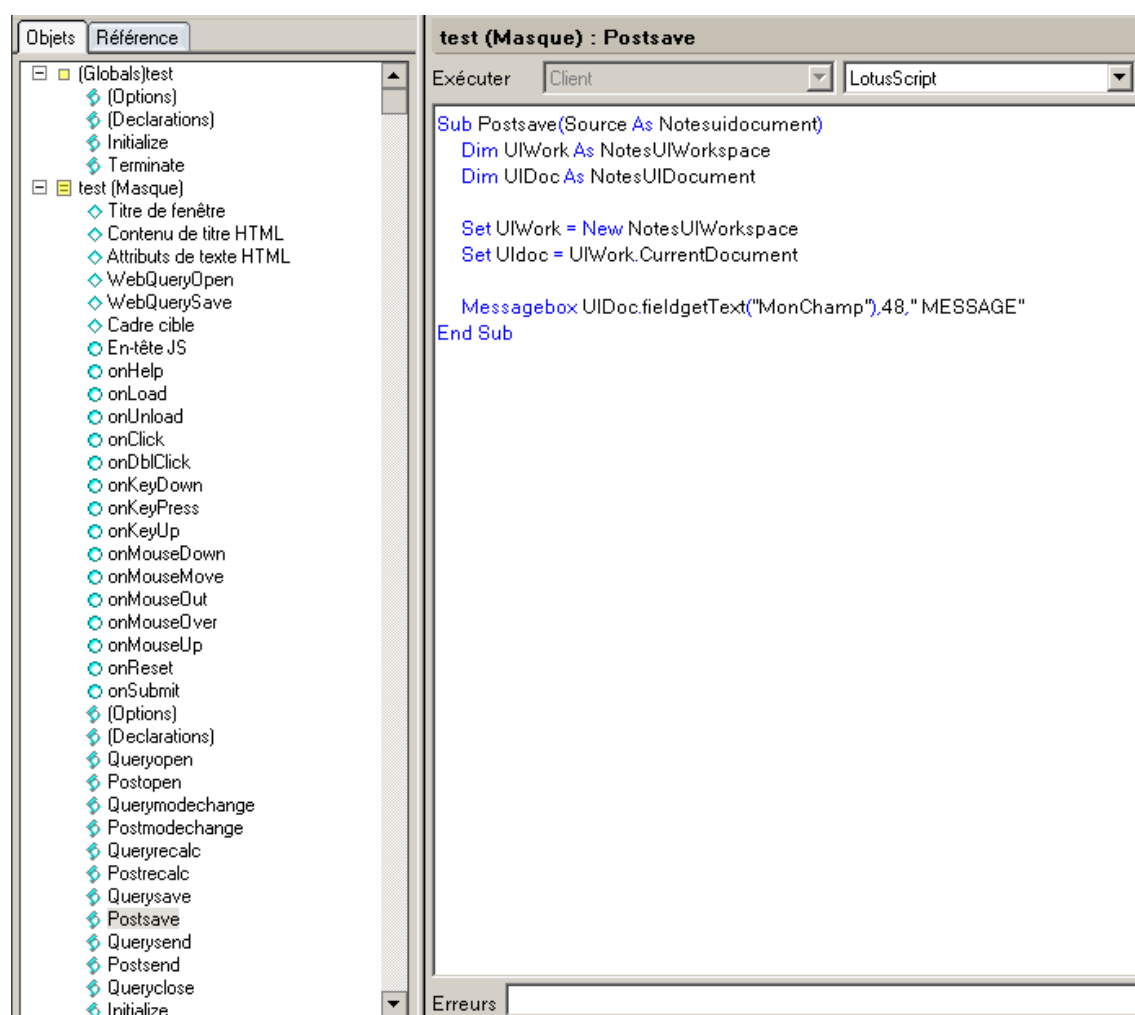
2 OÙ PUIS-JE UTILISER DU LOTUS SCRIPT ?

Tout comme pour les @Formules, vous pouvez utiliser le Lotus Script dans presque tous les éléments de design d'une application Lotus Notes (ou Base Notes). Vous pouvez aussi, ce qui n'est pas le cas en @Formules, créer des bibliothèques de fonctions ce qui vous permettront de « mutualiser » votre code (nous y reviendrons plus tard).

Le petit plus (absolument non négligeable) en ce qui concerne les éléments de design et le Lotus Script, c'est que la plupart de ces éléments comportent des procédures (ou Sub) natives à déclenchement événementiel (ex : à l'ouverture du masque, avant ou après l'enregistrement du masque...). Ces procédures comportent des variables pré-renseignées, utilisables immédiatement dans la procédure en cours. Il n'est donc pas nécessaire de les re-déclarer et de les ré-initialiser. Je donne un exemple ci-dessous, il est normal que vous ne compreniez pas grand-chose pour l'instant mais c'est pour vous montrer la différence (et par la suite vous faire gagner du temps dans vos développements).

En ouvrant les éléments dans le designer, vous verrez très vite les procédures dont je vous parle. Attention, il y a des procédures pour d'autres types de langage (java, java script, @formules).

Exemple de ce que l'on peut voir régulièrement dans des bases notes :



INITIATION AU LOTUS SCRIPT

Une procédure intégrée au masque qui se déclenche juste après l'enregistrement et qui affiche le contenu d'un champ du document.

```
Sub Postsave(Source As Notesuidocument)
  Dim UIWork As NotesUIWorkspace
  Dim UIDoc As NotesUIDocument

  Set UIWork = New NotesUIWorkspace
  Set UIDoc = UIWork.CurrentDocument

  MessageBox UIDoc.fieldgetText("MonChamp"),48," MESSAGE"
End Sub
```

Il n'est pas utile de créer et d'initialiser (instancier) la variable « Uidoc », elle est déjà connue et initialisée. Il s'agit de la variable « Source » contenue dans la déclaration de la procédure.

```
Sub Postsave(Source As Notesuidocument)

  MessageBox Source.fieldgetText("MonChamp"),48," MESSAGE"

End Sub
```

Plus rapide non ? Et bien plus fiable !

3 PRINCIPE DE BASE : LA DIFFERENCE ENTRE FRONTAL ET DORSAL

Lotus Notes n'étant pas un système de bases de données relationnel, mais un système documentaire, il a un principe de fonctionnement reposant sur une dualité : frontal/dorsal (front-end et back-end en anglais). Pour simplifier :

- Le frontal représente tout ce que vous voyez à l'écran (documents, vues...), ce sont les objets commençant par « NotesUI » (User interface) dans l'aide du Designer.
- La dorsal représente ce qui est en « mémoire » (documents, vues...) ce sont les objets commençant par « Notes » sans la notion de « UI » dans l'aide du Designer.

Vous retrouverez des objets similaires en frontal et en dorsal, mais pas avec les mêmes propriétés ni les mêmes méthodes. Il y a d'ailleurs bien plus d'objets dorsaux que frontaux.

ATTENTION

Il y a quelques difficultés liées à cette dualité.

Par exemple, les agents « schedulés » ne doivent jamais utiliser d'objets frontaux, ils ne fonctionnent pas.

Il faut éviter de mélanger programmation frontale et dorsale dans les masques (et dans les autres éléments en général) sous peine de voir certaines informations disparaître. Pour Lotus Notes, il y a deux objets bien distincts, l'un frontal l'autre dorsal et la « synchronisation » entre les deux assés subtile. Donc au début vous risquez d'avoir quelques surprises si vous faite le mélange, dans ce cas pensez à `NotesUIDocument.refresh` et `NotesUIDocument.reload` (pour plus d'info voir l'aide en ligne du designer).

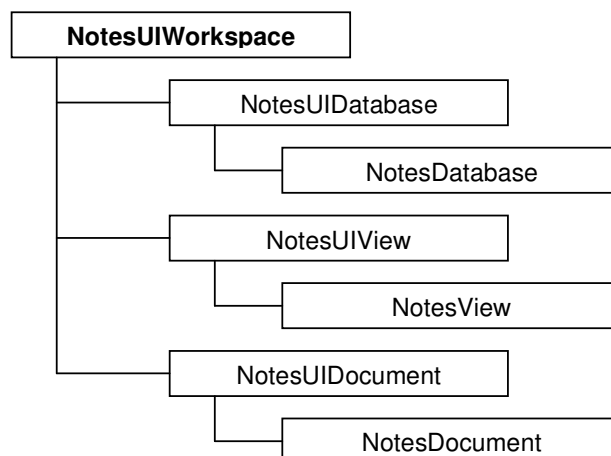
L'objet `NotesRichTextItem` pose quelques soucis en frontal, surtout avec les pièces jointes.

De même pour les champs créés en dorsal qui ne sont pas visibles dans les vue car la propriété « `IsSummary` » est à « `False` » au lieu de « `True` »

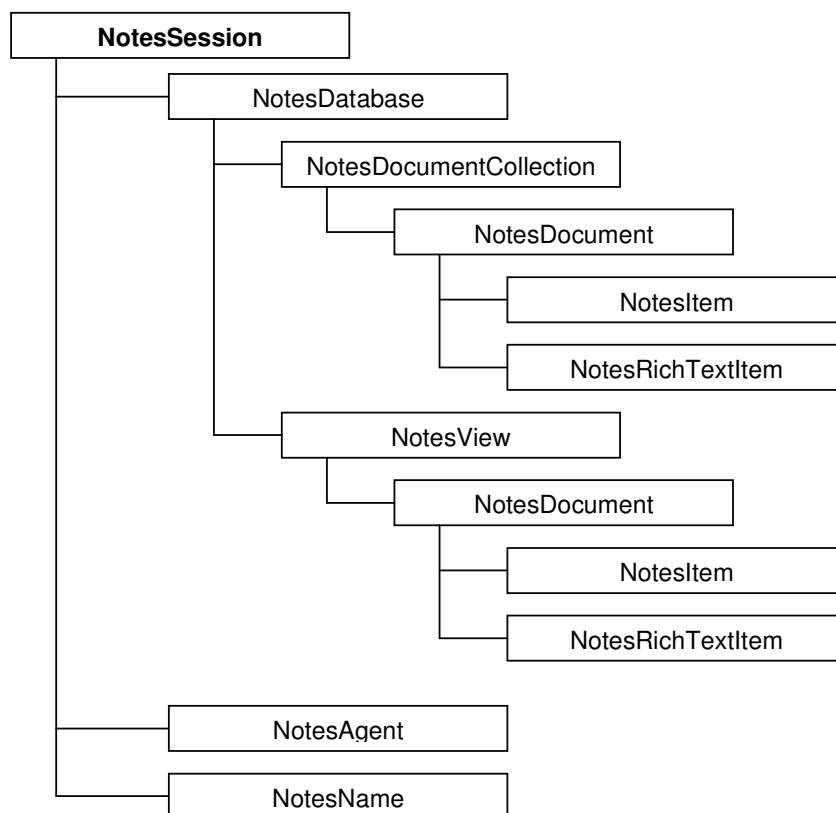
3.1 ARBORESCENCE DES PRINCIPAUX OBJETS

L'arborescence ne représente que les principaux objets, afin de permettre une meilleure visualisation de la hiérarchie entre les différents objets ainsi que de leurs dépendances.

3.1.1 CLASSES FRONTALES



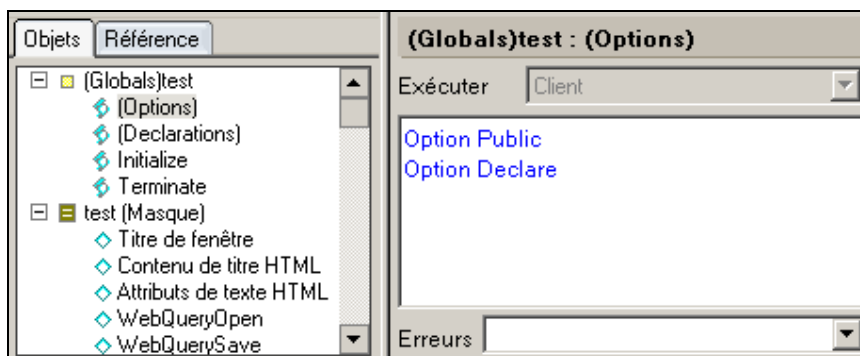
3.1.2 CLASSES DORSALES



Comme on peut le voir, il y a des classes dorsales dans l'arborescence frontale. Le mélange est voulu : l'on peut atteindre le dorsal via le frontal. Dans l'arborescence dorsale, l'objet **NotesDocument** (et les objets qui en dépendent) revient de manière récurrente pour bien montrer l'aspect « central » des documents pour Lotus Notes.

4 LES VARIABLES

Lotus Notes n'oblige pas à déclarer explicitement les variables mais il est vital de le faire. Vous éviterez beaucoup de soucis, de « bug » et votre maintenance sera plus facile. Pour cela il faut dans la partie « option » de votre élément de design rajouter : Option Declare



Pour obtenir la liste complète des variables, leurs définitions et comment les déclarer, regardez dans l'aide en ligne du designer. Mais vous utiliserez principalement :

- String : chaîne de caractères
- Integer ou long : numérique
- Variant : peut contenir n'importe quoi (très consommateur en ressources mémoires)
- Variable tableau : pour des tableaux à plusieurs dimensions mais un nombre de lignes fixe
- Variable liste : pour les tableaux à une seule dimension mais sans limitation sur le nombre de lignes

Déclaration d'une variable

Dim « nom_de_la_variable » as « type de la variable »

```
Dim stest As String
Dim i As Integer
Dim L As Long
Dim vrValue As Variant
```

ATTENTION

Vous avez la possibilité de déclarer les variables du même type à la suite :

```
Dim sText1, sText2, sText3 as string
```

A éviter, car si la première est bien déclarée en temps que chaîne de caractères les suivantes risquent d'être déclarées en temps que « variant » (ce qui consomme beaucoup plus de mémoire). Il est donc préférable de les déclarer comme cela :

```
Dim sText1 as string
Dim sText2 as string
Dim sText3 as string
```

J'ai pris comme exemple une variable de type string mais c'est valable pour tous les types.

POUR INFORMATION

Les variables et le Lotus Script en général (à l'opposé du Java et du Java Script) ne respectent pas la casse :

```
sText1 = stExt1  
Session = sesSion
```

Il est quand même préférable, dans la mesure du possible, de respecter la casse.

4.1 CONVERSION DES VARIABLES

Vous avez différentes commandes pour convertir vos variables d'un format vers un autre. Du texte vers du numérique et inversement. Vous trouverez dans l'aide en ligne du designer toutes les commandes. Elles commencent en général par la lettre « C » (cstr, cint, clng).

4.2 LA VARIABLE TABLEAU

4.2.1 DECLARATION D'UN TABLEAU

```
Dim « nom_de_la_variable » (dimension 1, dimension 2,...) as « type de la variable »
```

Une variable tableau peut être une variable de n'importe quel type :string, Integer, Long, NotesDocument, NotesView... cela offre bien des possibilités.

```
Dim atest (0 To 10, 1 To 4) As NotesDocument
```

C'est un tableau à deux dimensions, la première dimension va de zéro à dix et la deuxième va de un à quatre. Vous pouvez mettre autant de dimensions que vous voulez. Mais vous dépasserez rarement quatre dimensions.

Le dimensionnement d'un tableau peut être « partiellement » dynamique, j'utilise le mot « partiellement » car il y a certaines contraintes :

- Le tableau doit être déclaré de manière spécifique
- Vous ne pouvez pas modifier le nombre de dimensions mais juste leur taille

La déclaration d'un tableau dynamique est très simple : **Dim atest () as string**, comme vous pouvez le remarquer l'on n'indique aucune dimension. Il faut rajouter une ligne après pour le dimensionner **redim atest(0 to 10, 1 to 4)**. Vous pouvez aussi passer une variable numérique dans le dimensionnement.

```
Dim i As Integer  
i = 4  
Dim aTest () As NotesView  
Redim aTest (0 To i, 1 To 8)  
i = 20  
Redim aTest (1 To i , 0 To 15)
```

INITIATION AU LOTUS SCRIPT

Attention, quand vous redimensionnez un tableau, vous perdez toutes les informations qu'il contient. Pour éviter cela, il faut utiliser la commande « Preserve ».

```
Dim i As Integer
i = 4
Dim aTest () As NotesView
Redim aTest (0 To i, 1 To 8)
i = 20
Redim Preserve aTest (1 To i, 0 To 15)
```

ATTENTION

Il est très tentant d'utiliser la commande « Redim » dans une boucle, bien que cela soit très pratique, il faut savoir que c'est une commande qui consomme beaucoup de ressources système. Il faut donc l'utiliser judicieusement.

4.2.2 MANIPULATION D'UN TABLEAU

Renseigner un tableau

Il faut indiquer l'index de toutes les dimensions

```
Dim aTest (0 To 10, 1 To 4) As String

aTest(0,1) = "aaaa"
aTest(0,2) = "bbbb"
aTest(1,1) = "cccc"
aTest(1,2) = "dddd"
```

RECUPERER LA VALEUR D'UN ELEMENT

Il faut indiquer l'index de toutes les dimensions de la même manière.

```
Dim aTest (0 To 10, 1 To 4) As String
aTest(0,1) = "aaaa"
aTest(0,2) = "bbbb"
aTest(1,1) = "cccc"
aTest(1,2) = "dddd"

MessageBox aTest(0,1)
```

EFFACER LE CONTENU D'UN TABLEAU

Pour effacer le contenu il suffit d'utiliser la commande « erase »

INITIATION AU LOTUS SCRIPT

```
Dim aTest (0 To 10, 1 To 4) As String
```

```
aTest(0,1) = "aaaa"
```

```
aTest(0,2) = "bbbb"
```

```
aTest(1,1) = "cccc"
```

```
aTest(1,2) = "dddd"
```

```
Erase aTest
```

4.2.3 CONNAITRE LA TAILLE D'UN TABLEAU

Pour récupérer la taille d'une dimension d'un tableau il existe deux commandes :

- LBound qui donne la valeur basse
- Ubound qui donne la valeur haute

L'aide en ligne vous donnera toutes les informations sur ces deux commandes et les paramètres à passer pour récupérer la valeur d'une dimension particulière, si votre tableau est multidimensionnel.

```
Dim aTest (0 To 10, 1 To 4) As String
```

```
Dim I As Integer
```

```
Dim J As Integer
```

```
i = 0
```

```
aTest(i,1) = "aaaa"
```

```
aTest(i,2) = "bbbb"
```

```
i = i+1
```

```
aTest(i,1) = "cccc"
```

```
aTest(i,2) = "dddd"
```

```
For i = Lbound(aTest,1) To Ubound(aTest,1)
```

```
    For J = Lbound(aTest,2) To Ubound(aTest,2)
```

```
        Msgbox aTest(J,I)
```

```
    Next
```

```
Next
```

Autre méthode (cas d'un tableau à une seule dimension)

```
Dim aTest (0 To 3) As String
```

```
aTest(0) = "aaaa"
```

```
aTest(1) = "bbbb"
```

```
aTest(2) = "cccc"
```

```
aTest(3) = "dddd"
```

```
Forall sValue In aTest
```

```
    Msgbox sValue
```

```
End Forall
```

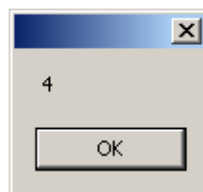
CONNAITRE LE NOMBRE D'ELEMENTS

```
Dim aTest (0 To 3) As String  
Dim i As Integer
```

```
aTest(0) = "aaaa"  
aTest(1) = "bbbb"  
aTest(2) = "cccc"  
aTest(3) = "dddd"
```

```
i = (Ubound(aTest)-Lbound(aTest))+1
```

```
Msgbox Cstr(i)
```



4.2.4 UTILISATION DE « OPTION BASE »

La directive Option Base permet de préciser si le tableau commence à l'indice 0 ou 1. Par défaut, la directive est fixée à Option Base 0. L'utilisation d'Option Base (ou du moins savoir comment est géré l'index) permet de déclarer un tableau d'une autre manière.

```
Dim aNom(5) As String  
Dim aAge(5) As Integer
```

Par conséquent Dim aNom(5) définit un tableau contenant 6 éléments de 0 à 5.
En revanche si Option base est à 1, ce tableau comporterait 5 éléments de 1 à 5.

4.3 LA VARIABLE LISTE

4.3.1 DECLARATION D'UNE LISTE

```
Dim « nom_de_la_variable » List as « type de la variable »
```

Tout comme la variable tableau, vous pouvez utiliser une variable de n'importe quel type :string, Integer, Long, NotesDocument, NotesView... Cela offre toujours bien des possibilités.

A la différence de la variable tableau dont le dimensionnement ne peut être que numérique, dans les variables liste, l'index peut contenir aussi du texte. On parle plus alors d'index mais de « TAG ».

INITIATION AU LOTUS SCRIPT

Attention si vous utilisez du texte dans votre tag, la variable liste respecte la case : Maison <> maison.

```
Dim LstTest List As String
```

4.3.2 MANIPULATION D'UNE LISTE

RENSEIGNER LA LISTE

Le fonctionnement est identique à un tableau à une dimension à la différence que le tag peut être alphanumérique. Vous pouvez bien sûr utiliser un variable comme Tag

```
Dim LstTest List As String  
Dim i As Integer
```

```
LstTest(0) = "aaa"  
LstTest(1) = "bbb"  
i=2  
LstTest(i) = "ccc"  
i=i+1
```

```
Dim LstTest2 List As String  
Dim sTag As String
```

```
LstTest2("a") = "aaa"  
LstTest2("A") = "bbb"  
sTag = "b"  
LstTest2(sTag) = "ccc"  
sTag = sTag+"1"  
LstTest2(sTag) = "ddd"
```

RECUPERER UN ELEMENT DE LA LISTE

Pour récupérer un élément de la liste, il faut faire référence à son tag. Il faut bien respecter le format et la case du tag. Si vous avez passé un chiffre sous la forme d'un integer vous devrez toujours utiliser ce format, vous ne pourrez pas l'appeler via une chaîne de caractères, pour résumer : 1 <> "1".

```
Dim LstTest List As String  
Dim i As Integer
```

```
LstTest(1) = "aaa"  
LstTest("1") = "bbb"
```

```
Msgbox LstTest(1)  
Msgbox LstTest("1")
```

Si votre variable liste est un objet (NotesDocument, NotesView...), vous devrez utiliser les mêmes méthodes d'initialisation utilisées par l'objet.

```
Dim LstTest List As NotesDocument  
Dim Doc As NotesDocument  
Dim Doc2 As NotesDocument
```

```
Set LstTest("Junior") = Doc  
Set LstTest ("Senior") = Doc
```

```
Set Doc2 = LstTest("Senior")
```

```
Dim LstTest List As NotesDocument  
Dim Doc As NotesDocument  
Dim Doc2 As NotesDocument  
Dim sText As String
```

```
sText = "Junior"  
Set LstTest(sText) = Doc  
sText = "Senior"  
Set LstTest (sText) = Doc
```

```
sText = "Junior"  
Set Doc2 = LstTest(sText)
```

INITIATION AU LOTUS SCRIPT

EFFACER LE CONTENU D'UNE LISTE

Pour effacer le contenu, il suffit d'utiliser la commande « erase »

```
Dim LstTest List As String

LstTest(0) = "aaa"
LstTest(1) = "bbb"
LstTest(2) = "ccc"
LstTest(3) = "ddd"

Erase LstTest
```

4.4 UN PETIT TRUC A PROPOS DU NOMMAGE DES VARIABLES

Hormis certains mots réservés, vous pouvez nommer vos variables comme bon vous semble. Je ne vous ferai pas un discours sur les noms trop longs ou trop courts, mais sur une astuce qui vous permettra de gagner du temps pour la maintenance et d'éviter des bugs stupides (qui nous arrivent à tous).

L'astuce consiste à préfixer le nom de sa variable en fonction de son type.

```
Dim i_compte As Integer
Dim L_compte2 As Long
Dim s_Text as string
Dim nm_User As NotesName
Dim Uldoc As NotesUIDocument
Dim vwtest As NotesView
```

L'intérêt est que par la suite, pendant la maintenance vous saurez tout de suite quel est le type de la variable sans regarder sa déclaration, ce qui est très pratique quand vous devez reprendre un code de 200 lignes. De plus, cela évite dès la saisie les erreurs de ce genre :

```
Dim i As String
Dim j As Integer
Dim K As Long

i = "aaa"
J = 123
K = J+i
```

Pour les même raisons, il est préférable de nommer les objets frontaux en les préfixant par UI, afin d'éviter toute erreur avec les objets dorsaux.

Il n'y a pas de règle pour le préfixe, mais vous vous ferez rapidement la votre si vous utilisez cette astuce. Tous les exemples de ce tutorial respectent une règle de nommage. Pour ma part, je l'utilise depuis longtemps et je l'applique aussi pour le nommage des champs et des pseudonymes des éléments de design. Je n'y ai trouvé que des avantages.

INITIATION AU LOTUS SCRIPT

5 ACCEDER A LA BASE NOTES EN COURS

Pour instancier la plupart des objets dont vous aurez besoin, vous devrez, au préalable, ouvrir une session et ouvrir la base de document en cours (c'est-à-dire celle sur laquelle vous travaillez)

5.1 DECLARATION ET OUVERTURE D'UNE SESSION

L'objet NotesSession vous donne entre autre accès à la base notes en cours et au nom de l'utilisateur en cours ainsi qu'à d'autres informations (pour plus d'informations sur cet objet voir l'aide en ligne du designer).

```
Dim Session As NotesSession 'déclaration  
Set Session = New NotesSession 'initialisation de la variable
```

5.2 DECLARATION ET OUVERTURE DE LA BASE LOTUS NOTES EN COURS

Pour accéder à la base Lotus Notes en cours, il faut utiliser une propriété de l'objet NotesSession. : Currentdatabase.

```
Dim Session As NotesSession  
Dim DB As NotesDatabase  
  
Set Session = New NotesSession  
Set DB = Session.currentdatabase
```

Il existe d'autres méthodes mais celle-ci est la plus simple et la plus rapide

L'objet NotesDatabase permet aussi de se connecter à d'autre bases Lotus Notes, de créer des documents et fournit toutes les informations sur la base de document à laquelle il fait référence (pour plus d'info voir l'aide en ligne du designer).

6 LA GESTION DES DOCUMENTS

6.1 LA DIFFERENCE ENTRE UN DOCUMENT ET UN MASQUE

On pourrait penser qu'un document est un masque et qu'un masque est un document et faire l'amalgame entre les deux. Un masque est une manière d'afficher un document à l'écran. Vous pouvez facilement changer le masque d'un document sans changer le contenu du document.

Un document est l'unité de référence de Lotus Notes, on peut y avoir accès en Lotus Script, mais l'utilisateur ne le verra qu'au travers d'un masque. Si un document possède un champ mais que le masque qui lui est attaché ne possède pas ce champ, le champ n'est pas effacé, il est tout simplement invisible, mais on peut y accéder par programmation.

L'objet qui gère les documents en dorsal se nomme : NotesDocument

L'objet qui gère les masques en dorsal se nomme : NotesForm

6.2 DECLARATION ET CREATION D'UN DOCUMENT

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Doc As Notesdocument

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Doc = DB.CreateDocument

If Doc Is Nothing Then
    'vérifie que la variable est renseignée
End If

If Not Doc Is Nothing Then
    'vérifie que la variable n'est pas renseignée
End If
```

6.3 MANIPULER LES CHAMPS D'UN DOCUMENT

RENSEIGNER UN CHAMP DU DOCUMENT

Deux méthodes identiques :

```
Doc.MonChamp = "aaaa"
Call Doc.ReplaceItemValue("MonChamp", "aaaa")
```

Si le champ n'existe pas, il est automatiquement créé en fonction du type d'informations enregistrées (chaîne de caractères, numérique...).

INITIATION AU LOTUS SCRIPT

RECUPERER LA VALEUR D'UN CHAMP

Deux méthodes identiques :

```
Dim sTest As String  
sTest = Doc.MonChamp(0)  
sTest = Doc.GetItemValue("MonChamp")(0)
```

Pour information

Les deux méthodes pour renseigner et récupérer la valeur d'un champ sont identiques, vous pouvez les utiliser indifféremment. Mais les méthodes « ReplaceItemValue » et « GetItemValue » sont un tout petit peu plus rapides (la différence n'est visible que sur des milliers de traitements)

6.4 LE MASQUE AFFECTE AU DOCUMENT

Comme l'on peut affecter n'importe quel masque à un document, il existe un champ réservé qui contient le nom du masque à afficher. Nom du champ : « Form ». Ce champ se gère comme n'importe quel autre champ. Si vous créez un document via du Lotus Script en dorsal vous devez renseigner ce champ.

```
Dim Session As NotesSession  
Dim DB As NotesDatabase  
Dim Doc As Notesdocument  
  
Set Session = New NotesSession  
Set DB = Session.currentdatabase  
Set Doc = DB.CreateDocument  
Doc.form = "memo"
```

6.5 LES COMMANDES DE BASES D'UN DOCUMENT

ENREGISTRER UN DOCUMENT

```
Dim Session As NotesSession  
Dim DB As NotesDatabase  
Dim Doc As Notesdocument  
  
Set Session = New NotesSession  
Set DB = Session.currentdatabase  
Set Doc = DB.CreateDocument  
Doc.form = "memo"  
Call Doc.save(False,False)
```

INITIATION AU LOTUS SCRIPT

EFFACER UN DOCUMENT

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Doc As Notesdocument

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Doc = DB.CreateDocument
Doc.form = "memo"
Call Doc.save(True,False)
Call doc.remove(True)
```

ENVOYER UN DOCUMENT COMME UN MAIL

Il existe plusieurs variantes pour envoyer un mail via du Lotus Script. En dorsal, elles s'appuient toutes sur la méthode « Send » de l'objet « NotesDocument ».

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Doc As Notesdocument

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Doc = DB.CreateDocument
Doc.form = "memo"
Doc.Subject = "Sujet du mail"
Doc.Body = "Corps du mail"
Call Doc.send(False,session.username)
```

6.6 IDENTIFIANT UNIQUE & IDENTIFIANT UNIVERSEL

Chaque document possède un identifiant unique (NotesID) et un identifiant Universel (UniversalID). Ces deux identifiants sont uniques pour chaque document (ils changent en cas de copier/coller). Ils permettent d'identifier précisément un document (à la manière d'une clé unique). Ils sont automatiquement générés.

NotesID : Combinaison alphanumérique sur 8 caractères maximum (Donc la combinaison peut contenir moins de 8 caractères), représente le document uniquement à l'intérieur de la base Notes dans laquelle il se trouve.

```
Dim Doc As NotesDocument
Dim sTest As String
stest = Doc.NotesID
```

INITIATION AU LOTUS SCRIPT

UniversalID : Combinaison alphanumérique de 32 caractères (l'universalID est TOUJOURS composé de 32 caractères), fait référence à un seul document (dans une base Notes bien précise) quelle que soit la base Notes dans laquelle on se trouve, ce qui est très pratique dans le cas de bases notes répliquées (pour plus d'information sur la réplication des base Notes voir l'aide en ligne du designer).

```
Dim Doc As NotesDocument
Dim sTest As String
sTest = Doc.UniversalID
```

INITIALISER UN DOCUMENT VIA SON ID

Pour retrouver un document en fonction de son Id vous avez deux méthodes de l'objet NotesDatabase :

- GetDocumentById (pour retrouver le document par le NotesID)
- GetDocumentByUNID (pour retrouver le document par son UniversalID)

L'on considère que dans l'exemple suivant, la variable DocID contient le NotesID et la variable DocUNID contient l'UniversalID du document à retrouver.

```
Dim Session as NotesSession
Dim DB as NotesDatabase
Dim Doc1 as NotesDocument
Dim Doc2 as NotesDocument
Dim sDocID as String
Dim sDocUNID as String

Set Session = New NotesSession
Set DB = Session.currentdatabase

sDocID = "8FA"
sDocUNID = "1239CD8106EC0F93C125702B0030BCAC"

Set Doc1 = DB.GetDocumentById(sDocID)
Set Doc2 = DB. GetDocumentByUNID(sDocUNID)
```

ATTENTION

Si le NotesID ou l'UniversalID n'est pas valide, Lotus Notes générera une erreur.

6.7 LA GESTION DES CHAMPS DANS UN DOCUMENT

LA PARTICULARITE DES CHAMPS LOTUS NOTES

Les champs sont gérés par Lotus Notes comme des variables tableau à une dimension dont l'index commence à zéro. L'ont parle généralement de champ « Mono-Valué » quand le champ ne comporte qu'une seule valeur/Ligne et de champ « Multi-Valué » quand le champ/tableau comporte

INITIATION AU LOTUS SCRIPT

plusieurs valeurs/lignes. (Pour le paramétrage des champs Multi-valués voir l'aide en ligne du designer). La où les valeurs contenues dans un champ peuvent être de types numériques, chaîne de caractère, date...

Donc pour récupérer la valeur d'un champ, il faut indiquer son nom et l'index :

Champ Mono-Valué, c'est à dire les champs qui ne contiennent qu'une seule valeur et non une liste de valeurs. Ces champs sont considérés comme un tableau à une dimension avec une seule ligne. Pour récupérer la valeur du champ, il suffit de lire la référence à l'index zéro. (Par contre pour renseigner un champ Mono-Valué vous n'avez pas besoin d'indiquer son index)

```
Doc.MonChamp = "AAAA"  
Msgbox Doc.MonChamp(0)
```

Champ Multi-Valué, c'est à dire les champs contenant une liste de valeurs. Ces champs sont gérés comme un tableau à une dimension avec une ligne par valeur contenue dans le champ. Donc, pour récupérer une de ces valeurs, il faut faire référence à l'index de cette valeur. Attention, si vous faites référence à un index qui n'existe pas (ex : le champ contient trois valeurs et vous faites référence à une quatrième) Lotus Notes générera une erreur. L'index commence toujours à zéro.

A la différence d'un tableau vous ne pouvez pas renseigner une valeur via son index : Doc.MonChamp(3) = « aaa » renverra toujours un erreur. Pour renseigner un champ multi-valué vous devez renseigner les différentes valeur qui le composent l'une après l'autre ou passer directement la liste des valeurs.

Si vous utilisez une variable tableau, celui-ci doit être à une seule dimension et sont index doit commencer à zéro.

Si vous utilisez une variable liste, le tag doit être numérique et lui aussi commencer à zéro.

L'exemple suivant renseigne un champ multi-valué via une variable tableau

```
Dim aTableau(0 To 3)  
aTableau(0) = "aaaa"  
aTableau(1) = "bbbb"  
aTableau(2) = "cccc"  
aTableau(3) = "dddd"  
  
Doc.MonChamp = aTableau  
  
aTableau(1) = "1111"  
  
Doc.MonChamp = aTableau
```

6.7.1 L'OBJET « NOTESITEM »

Pour gérer les champs d'un document il existe un objet : NotesItem qui représente un champ dans un document. Cet objet permet de récupérer la ou les valeurs d'un champ et toutes les informations relatives à ce champ.

INITIATION AU LOTUS SCRIPT

INITIALISER UN OBJET NOTESITEM

Il existe plusieurs méthodes, voici la plus simple (si le champ existe) :

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Doc As NotesDocument
Dim Item As NotesItem

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Doc = DB.CreateDocument
Doc.form = "memo"
Doc.test = "aaa"

Set Item = Doc.GetfirstItem("test")
```

Si le champ n'existe pas :

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Doc As NotesDocument
Dim Item As NotesItem

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Doc = DB.CreateDocument
Doc.form = "memo"

Set Item = Doc.GetfirstItem("test")
If item Is Nothing Then
    Set item = New NotesItem(Doc,"test","aaa")
    'Vous aurez toutes les informations dans l'aide en ligne du designer
End If
```

PASSER EN REVUE TOUTES LES VALEURS D'UN CHAMP MULTI-VALUE

```
Set Item = Doc.GetfirstItem("test")
If item Is Nothing Then
    Forall Value In Item.Values
        MsgBox Cstr(value)
    End Forall
End If
```

INITIATION AU LOTUS SCRIPT

RAJOUTER UNE VALEUR A LA FIN D'UN CHAMP MULTI-VALUE

```
Doc.Test = "aaaa"  
Set Item = Doc.GetfirstItem("Test")  
Call Item.AppendToTextList("bbbb")
```

SUPPRIMER DEFINITIVEMENT UN CHAMP DANS UN DOCUMENT

```
Dim Session As NotesSession  
Dim DB As NotesDatabase  
Dim Doc As NotesDocument  
Dim Item As NotesItem  
  
Set Session = New NotesSession  
Set DB = Session.currentdatabase  
Set Doc = DB.CreateDocument  
Doc.form = "memo"  
Doc.test = "aaa"  
  
Set Item = Doc.GetfirstItem("test")  
Call Item.remove
```

6.8 LA GESTION DES CHAMPS DE TYPE « TEXT RICHE »

La gestion des champs de type « Texte Riche » ou « Rich Text » nécessite une bonne connaissance du Lotus Script et de Lotus Notes en général.

Un champ « Rich Text » est un champ qui peut contenir n'importe quoi et surtout qui permet de faire de la mise en page à l'intérieur (changer la taille de la police, sa couleur, générer des tableaux, insérer des pièces jointes...). Le meilleur exemple de « Rich Text » est le corps d'un mail, vous pouvez y faire ce que vous voulez, et bien le corps d'un mail est en fait un champ « Rich Text » (son nom dans le masque est « Body »).

Il existe plusieurs objets permettant de gérer ce type de champ, ici nous n'en verrons qu'un seul, le principal : NotesRichTextItem.

6.8.1 DECLARATION ET INITIALISATION

Un champ un « Rich Text » s'initialise comme un champ normal.

```
Dim Session As NotesSession  
Dim DB As NotesDatabase  
Dim Doc As NotesDocument  
Dim rItem As NotesRichTextItem  
  
Set Session = New NotesSession  
Set DB = Session.Currentdatabase  
Set Doc = DB.GetDocumentByID("8F45")  
  
Set ritem = Doc.GetFirstItem("Body")
```

INITIATION AU LOTUS SCRIPT

6.8.2 MANIPULATION D'UN CHAMP « RICH TEXT »

CREER LE CHAMP S'IL N'EXISTE PAS

```
Set rItem = Doc.GetFirstItem("Body")
If rItem Is Nothing Then
    'création du champ s'il n'existe pas
    Set rItem = New NotesRichTextItem( Doc, "Body" )
End If
```

INSERER UN TEXTE DANS LE CHAMP

```
Set rItem = Doc.GetFirstItem("Body")
Call rItem.AppendText("aaaa")
```

AJOUTER UN TEXTE A LA FIN DU CHAMP

```
Set rItem = Doc.GetFirstItem("Body")
Call rItem.AppendText("aaaa")
Call rItem.AppendTextList("bbbb")
```

6.8.3 INSERER UN « LIEN DOC » DANS UN CHAMP « RICH TEXT »

L'objet possède plusieurs autres méthodes, pour ajouter des sauts de lignes, des tabulations... l'aide en ligne du designer vous les fournira toutes. Une des méthodes les plus intéressantes est celle qui permet d'ajouter un « DocLink », un raccourci notes vers un document. Il apparaît sous la forme d'une petite icône. Très pratique dans les envois de mails automatiques pour pointer directement vers le bon document.

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Doc As NotesDocument
Dim Doc2 As NotesDocument
Dim rItem As NotesRichTextItem

Set Session = New NotesSession
Set DB = Session.CurrentDatabase
Set Doc = DB.GetDocumentByID("8F45")
Set Doc2 = DB.GetDocumentByID("6645D4")
Set rItem = New NotesRichTextItem( Doc, "Body" )
Call rItem.AppendText("lien vers le document de référence => ")
Call rItem.AppendDocLink(Doc2,"Lien Document")
```

7 LES VUES & LES COLLECTIONS

7.1 LES VUES

Les vues servent à afficher, sélectionner trier et/ou catégoriser les documents d'une base notes. L'on a régulièrement besoin de parcourir les documents se trouvant dans une vue. Les vues sont gérées par l'objet NotesView.

7.1.1 INITIALISER UN OBJET NOTESVIEW

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim vwVue As NotesView

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set vwVue = DB.GetView("Nom_De_Ma_Vue")
```

7.1.2 MANIPULATION D'UNE VUE

Les documents sont triés dans l'objet NotesView comme ils sont triés à l'affichage.

PARCOURIR LA VUE POUR PASSER EN REVU TOUS LES DOCUMENTS

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim vwVue As NotesView
Dim Doc As NotesDocument

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set vwVue = DB.GetView("Nom_De_Ma_Vue")
If Not vwVue Is Nothing Then
  Set Doc = vwVue.GetFirstDocument
  While Not Doc Is Nothing
    MsgBox doc.universalid
    Set Doc = vwVue.GetNextDocument(Doc)
  Wend
End If
```

La boucle fonctionne tant que la fonction « Next » trouve un document. Le premier « set Doc » sert à initialiser la variable Doc avec le premier document de la vue.

INITIATION AU LOTUS SCRIPT

RAFRAICHIR LE CONTENU D'UNE VUE

Vous aurez parfois besoin de rafraîchir le contenu d'une vue, du moins de l'objet NotesView pour être certain que les nouveaux documents y sont bien pris en compte :

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim vwVue As NotesView

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set vwVue = DB.GetView("Nom_De_Ma_Vue")

If Not vwVue Is Nothing Then
    Call vwVue.Refresh
End If
```

7.1.3 RECHERCHER UN DOCUMENT DANS UNE VUE EN FONCTION D'UN MOT CLE

Vous pouvez faire l'équivalent de la @formule « @Dblookup » en Lotus Script. Comme pour le @Dblookup la recherche se fait sur la première colonne de la vue et cette colonne doit être triée. Il y a deux méthodes de l'objet NotesView qui le permettent :

- GetDocumentByKey : permet de récupérer un document et un seul.
- GetAllDocumentsByKey : permet de récupérer une collection de documents (correspondant au critère de recherche)

Exemple N° 1 : Récupère le premier document correspondant au critère.

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim vwVue As NotesView
Dim Doc As NotesDocument

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Vue = DB.GetView("Mavue")

Set Doc = vwVue. GetDocumentByKey("aaa", True)
```

Exemple N° 2 : Récupère tous les documents correspondant au critère.

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim vwVue As NotesView
Dim Collection As NotesDocumentCollection

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set vwVue = DB.GetView("Mavue")

Set Collection = vwVue.GetAllDocumentsByKey("aaa", True)
```

7.2 LES COLLECTIONS DE DOCUMENTS

Il s'agit d'un regroupement de documents, en fonction de critères contenus dans une base notes. Ils sont gérés par l'objet NotesDocumentCollection. Il y a plusieurs manières de générer une collection. Voici les deux principales, ce sont des méthodes de l'objet NotesDatabase : « Search » et « FTSearch ».

Un « Search » utilise le même type de sélection qu'une vue alors qu'un « FTSearch » permet la recherche « Full-Text ».

7.2.1 INITIALISATION D'UNE COLLECTION

INITIALISATION VIA UN « SEARCH »

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Collection As NotesDocumentCollection
Dim sCritere As String

Set Session = New NotesSession
Set DB = Session.currentdatabase

sCritere = {form = memo & @contains(test;"aaa")}
Set Collection = DB.Search(sCritere,Nothing,0)
```

INITIALISATION VIA UN « FTSEARCH »

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Collection As NotesDocumentCollection
Dim sCritere As String

Set Session = New NotesSession
Set DB = Session.currentdatabase

sCritere = Inputbox("Saisir la valeur à rechercher :", " SELECTION")
Set Collection = DB.FTSearch(sCritere, 0)
```

7.2.2 MANIPULATION D'UNE COLLECTION DE DOCUMENTS

VERIFIER QUE LA COLLECTION CONTIENT AU MOINS UN DOCUMENT

Pour vérifier qu'une collection contient au moins un document il faut faire une double vérification, car parfois, si le critère de recherche ne correspond à aucun document au lieu d'avoir une collection avec zéro document l'objet NotesDocumentCollection n'est pas initialisé.

```
If Not Collection Is Nothing Then
    If Collection.Count > 0 Then

        End If
    End If
```

INITIATION AU LOTUS SCRIPT

PARCOURIR LA COLLECTION POUR PASSER EN REVU TOUS LES DOCUMENTS

Le principe est le même que pour les vues :

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Collection As NotesDocumentCollection
Dim Doc As NotesDocument
Dim sCritere As String

Set Session = New NotesSession
Set DB = Session.currentdatabase

sCritere = "aaa"
Set Collection = DB.Search(sCritere,Nothing,0)

If Not Collection Is Nothing Then
  If Collection.Count > 0 Then
    Set Doc = Collection.GetFirstDocument
    While Not Doc Is Nothing
      MsgBox doc.universalid
      Set Doc = Collection.GetNextDocument(Doc)
    Wend
  End If
End If
```

Il existe une autre méthode qui semble plus pratique mais qui a un défaut, sa consommation en ressources système (celle-ci est exponentielle par rapport au nombre de document). Il convient donc de l'utiliser judicieusement car elle peut considérablement rallonger les temps de traitement.

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Collection As NotesDocumentCollection
Dim Doc As NotesDocument
Dim sCritere As String
Dim i As Integer

Set Session = New NotesSession
Set DB = Session.currentdatabase

sCritere = {form = memo & @contains(test;"aaa")}
Set Collection = DB.Search(sCritere,Nothing,0)

If Not Collection Is Nothing Then
  If Collection.Count > 0 Then
    For I = 1 To Collection.count
      Set Doc = Collection.GetNTHDocument(i)
      MsgBox doc.universalid
    Next
  End If
End If
```

ATTENTION

Si la base sur laquelle est fait le "Search" ou le "FTSearch" est indexée, ces deux méthodes ne pourront pas retrouver un document tant que celui-ci ne sera pas indexé.

7.3 DIFFERENCE ENTRE UN « SEARCH » ET UN « GETALLDOCUMENTBYKEY »

Comme vous avez remarqué, il existe principalement deux méthodes pour retrouver des documents :

- le DB.Search (ainsi que le DB.FTSearch)
- Le GetAllDocumentbyKey (ainsi que le GetDocumentByKey)

Ces deux méthodes qui donnent un résultat identique n'ont pas le même impact sur le système. Le DB.Search est plus long à fournir un résultat car il doit effectuer la recherche sur l'ensemble de la base notes. Le GetAllDocumentbyKey est plus rapide car il fait sa recherche sur une vue qui sélectionne déjà les documents.

Première réaction, je n'utilise pas le DB.Search, seulement il y a un « Mais », et de taille. Une vue ne consomme pas de mémoire, ni de CPU, elle consomme de l'espace sur le disque dur (du serveur en général) et de ce point de vue, une vue peut être très gourmande, même horriblement gourmande. Faire une vue pour chaque recherche peut faire exploser la taille de votre base Notes. De plus, la formule de sélection d'une vue n'est pas dynamique, impossible de passer une donnée extérieure via des @formules du genre @environment, @DBColumn, @DBLookup, @GetProfilField... et la sélection en fonction du nom d'un utilisateur (@username) n'est possible que dans les vues privées. Le DB.Search n'a pas ce genre de problème.

La solution : une bonne analyse du besoin pour une utilisation adaptée.

Si votre traitement est régulier (un agent programmé par exemple) et que la rapidité prime, vous vous tournerez plutôt vers une vue et le GetAllDocumentbyKey

Si votre traitement est exceptionnel, qu'il nécessite une formule de sélection complexe (voir irréalisable dans une vue) ou pour économiser de l'espace disque, vous utiliserez de préférence le DB.Search.

C'est très schématique, l'utilisation de l'un ou de l'autre dépendra de plusieurs facteurs que vous seul pouvez connaître. Une utilisation judicieuse de ces deux méthodes en fonction des caractéristiques de votre projet est impérative et vous sera profitable sur bien des points.

8 GESTION DES NOMS NOTES

Comme vous avez pu le remarquer le nom d'un utilisateur de Lotus notes respecte un certain formatage du genre Marc DUPONT/TEST/DEV. Les noms notes sont gérés par l'objet NotesName, c'est l'équivalent à la @formule : @name. Il existe trois formats principaux pour les noms Notes :

- Common = Marc DUPONT
- Abbreviated => Marc DUPONT/TEST/DEV
- Canonical => CN=Marc DUPONT/OU=TEST/OU=DEV

8.1 INITIALISER UN OBJET NOTESNAME

```
Dim Session As NotesSession
Dim nmUser As NotesName

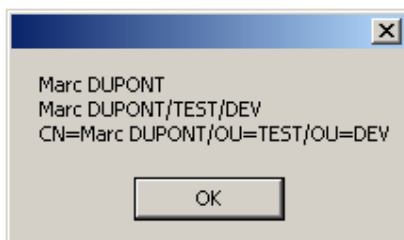
Set Session = New NotesSession
Set nmUser = New NotesName(Session.Username)
```

8.2 MANIPULER L'OBJET NOTESNAME

```
Dim Session As NotesSession
Dim nmUser As NotesName

Set Session = New NotesSession
Set nmUser = New
NotesName(Session.Username)

Msgbox (nmUser.Common) _
+ Chr(10)+(nmUser.Abbreviated)_
+ Chr(10)+nmUser.Canonical
```



Cet objet vous permettra de renseigner les champs de type « nom » de manière correcte ou de les afficher dans le format adéquat.

8.3 A PROPOS DES CHAMPS DE TYPE NOMS

On renseigne régulièrement les champs de type nom (name, reader et author) en dorsal. Il faut prendre l'habitude en dorsal de toujours renseigner ces champs avec le nom sous sa forme canonique. Pourquoi ? C'est une limitation de Notes (certains diront un bug). En dorsal pure, (dans un agent schedulé par exemple), seuls les noms canoniques sont pris en compte pour la gestion des accès au document via les champs de type auteur ou lecteur. Si vous renseignez le champ avec un nom abrégé cela n'a aucun effet. Le champ sera bien renseigné mais l'utilisateur n'aura pas d'accès. La solution la plus simple : toujours renseigner les champs de type nom avec la forme canonique.

9 LES AGENTS

Les agents sont des « robots » capables d'exécuter des instructions, Ils peuvent être lancés de manière manuelle, « schedulé » ou programmé. Nous nous intéresserons ici au lancement, programmé, via du Lotus Script d'un agent grâce à l'objet « NotesAgent ».

9.1 INITIALISATION DE L'AGENT

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Agent As NotesAgent

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Agent = DB.getAgent("Nom_De_Mon_agent")
```

9.2 LANCEMENT DE L'AGENT

Pour lancer un agent via du Lotus Script il faut la méthode : run

Call Agent.run

La méthode « run » permet de passer un objet NotesDocument en référence via son NoteID

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Agent As NotesAgent

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Agent = DB.getAgent("Nom_De_Mon_agent")

Call Agent.run
```

9.3 PASSER UN DOCUMENT EN PARAMETRE

Il est possible de passer un document en paramètre à un agent lors de son lancement via la NoteID du document.

INITIATION AU LOTUS SCRIPT

Lancer l'agent et passer le document en paramètre :

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Agent As NotesAgent
Dim Doc As NotesDocument

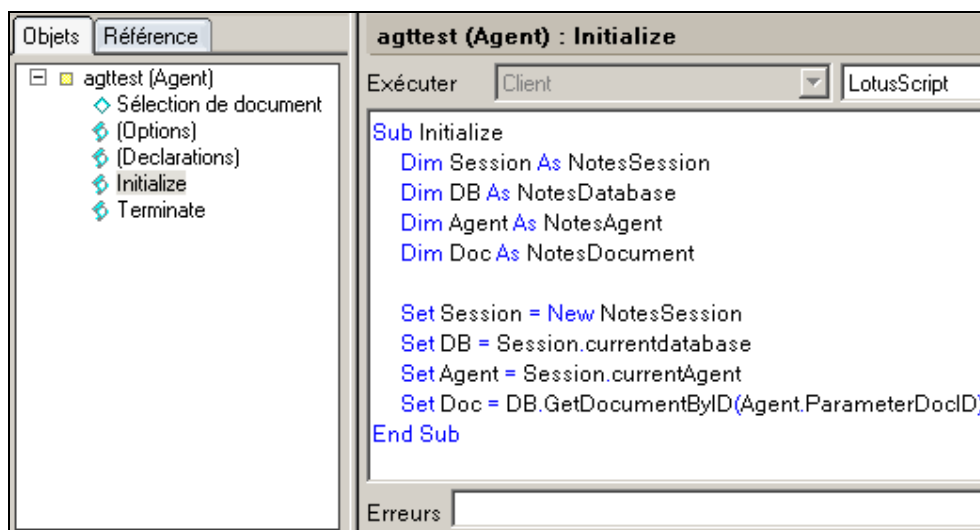
Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Agent = DB.getAgent("Nom_De_Mon_agent")
Set Doc = DB.GetDocumentById("1239CD8106EC0F93C125702B0030BCAC")

Call Agent.run(Doc.NoteID)
```

Dans le module « Initialize », il faut que l'agent se « connecte à lui-même » pour pouvoir récupérer le paramètre.

```
Sub Initialize
    Dim Session As NotesSession
    Dim DB As NotesDatabase
    Dim Agent As NotesAgent
    Dim Doc As NotesDocument

    Set Session = New NotesSession
    Set DB = Session.currentdatabase
    Set Agent = Session.currentAgent
    Set Doc = DB.GetDocumentById(Agent.ParameterDocID)
End Sub
```



Un agent permet aussi de travailler sur les documents sélectionnés dans une vue. Pour cela, il faut que les documents ciblés soient « les documents sélectionnés ». Il faut bien sûr pour cela que l'agent soit lancé depuis la vue (via un bouton).

INITIATION AU LOTUS SCRIPT

```
Sub Initialize
  Dim Session As NotesSession
  Dim DB As NotesDatabase
  Dim Collection As notesDocumentCollection

  Set Session = New NotesSession
  Set DB = Session.currentdatabase
  Set Collection = DB.UnprocessedDocuments
End Sub
```

10 PROGRAMMATION FRONTALE

10.1 L'OBJET NOTESUIWORKSPACE

L'objet NotesUIWorkSpace permet d'accéder au document et à la vue en cours, d'éditer des documents, d'afficher des boîtes de dialogue simples ou complexes. Il est très utilisé et l'aide en ligne du designer donne toutes les informations sur ses méthodes et propriétés.

10.1.1 INITIALISER L'OBJET NOTESUIWORKSPACE

```
Dim UIWork As NotesUIWorkSpace
Set UIWork = New NotesUIWorkSpace
```

10.1.2 MANIPULER L'OBJET NOTESUIWORKSPACE

ACCEDER AU DOCUMENT EN COURS (AFFICHE).

```
Dim UIWork As NotesUIWorkSpace
Dim UIDoc As NotesUIDocument

Set Uiwork = New NotesUIWorkSpace
Set UIDoc = UIWork.CurrentDocument
```

ACCEDER A LA VUE EN COURS (AFFICHEE).

```
Dim UIWork As NotesUIWorkSpace
Dim UIVue As NotesUIView

Set UIWork = New NotesUIWorkSpace
Set UIVue = UIWork.CurrentView
```

EDITER UN DOCUMENT.

```
Dim UIWork As NotesUIWorkSpace
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Doc As NotesDocument
Dim UIDoc As NotesUIDocument

Set Uiwork = New NotesUIWorkSpace
Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Doc = DB.CreateDocument
Doc.form = "memo"

Set UIDoc = UIWork.CurrentDocument(,True,Doc,False)
```

INITIATION AU LOTUS SCRIPT

INITIALISER UN OBJET NOTESUIDOCUMENT

```
Dim UIWork As NotesUIWorkspace  
Dim UIDoc As NotesUIDocument  
Dim Doc As NotesDocument  
  
Set Uiwork = New NotesUIWorkspace  
Set UIDoc = Uiwork.CurrentDocument
```

INITIALISER UN OBJET NOTESDOCUMENT

```
Dim UIWork As NotesUIWorkspace  
Dim UIDoc As NotesUIDocument  
Dim Doc As NotesDocument  
  
Set Uiwork = New NotesUIWorkspace  
Set UIDoc = Uiwork.CurrentDocument  
Set Doc = UIDoc.document
```

INITIALISATION DIRECT UN OBJET NOTESDOCUMENT

```
Dim UIWork As NotesUIWorkspace  
Dim UIDoc As NotesUIDocument  
Dim Doc As NotesDocument  
  
Set Uiwork = New NotesUIWorkspace  
Set Doc = Uiwork.CurrentDocument.document
```

Les méthodes d'initialisation présentées fonctionnent aussi pour les vues (NotesUIView, NotesView) et les Bases Notes (NotesUIDatabase, NotesDatabase). L'aide en ligne vous fournira plus d'informations à ce sujet.

10.1.3 LES BOITES DE DIALOGUE

Vous avez plusieurs possibilités pour faire des boites de dialogue, vous pouvez utiliser (entre autres) les méthodes « DialogBox », « PickListCollection » et « Prompt » :

« Prompt » : elle permet d'afficher différentes boites de dialogue.

« PickListCollection » : elle affiche une boite de dialogue qui permet de faire la sélection d'un ou plusieurs documents dans une vue.

« DialogBox » : elle permet d'afficher dans une boite de dialogue un masque ou un sous-masque, ce qui est très pratique pour faire des boites de dialogue très spécifiques.

L'aide en Ligne du designer vous donnera toutes les explications et exemples nécessaires.

10.2 L'OBJET NOTESUIDOCUMENT

C'est objet permet de gérer le document affiché à l'écran. C'est l'équivalent frontal de l'objet NotesDocument.

INITIATION AU LOTUS SCRIPT

ACCEDER AU DOCUMENT EN COURS (AFFICHE).

```
Dim UIWork As NotesUIWorkSpace
Dim UIDoc As NotesUIDocument

Set Uiwork = New NotesUIWorkSpace
Set UIDoc = UIWork.CurrentDocument
```

RENSEIGNER UN CHAMP DU DOCUMENT.

```
Dim UIWork As NotesUIWorkSpace
Dim UIDoc As NotesUIDocument

Set Uiwork = New NotesUIWorkSpace
Set UIDoc = UIWork.CurrentDocument

Call UIDoc.FieldSetText("Test", "aaa")
```

AJOUTER UNE VALEUR A UN CHAMP DU DOCUMENT.

```
Dim UIWork As NotesUIWorkSpace
Dim UIDoc As NotesUIDocument

Set Uiwork = New NotesUIWorkSpace
Set UIDoc = UIWork.CurrentDocument

Call UIDoc.FieldSetText("Test", "aaa")
Call UIDoc.FieldAppendText("Test", "bbb")
```

RECUPERER LA VALEUR D'UN CHAMP .

```
Dim UIWork As NotesUIWorkSpace
Dim UIDoc As NotesUIDocument
Dim stext As String

Set UIWork = New NotesUIWorkSpace
Set UIDoc = UIWork.CurrentDocument

sText = UIDoc.fieldgetText("test")
Msgbox stext
```

INITIATION AU LOTUS SCRIPT

POSITIONNER LE CURSEUR SUR UN CHAMP SPECIFIQUE

```
Dim UIWork As NotesUIWorkSpace  
Dim UIDoc As NotesUIDocument  
  
Set Uiwork = New NotesUIWorkSpace  
Set UIDoc = UIWork.CurrentDocument  
  
Call UIDoc.GotoField("Test")
```

ATTENTION

Alors qu'un document dorsal gère parfaitement les champs inexistants, pour un document frontal le champ doit exister dans le masque sinon Lotus Notes générera une erreur indiquant que le champ est introuvable

11 LES BIBLIOTHEQUES DE SCRIPTS

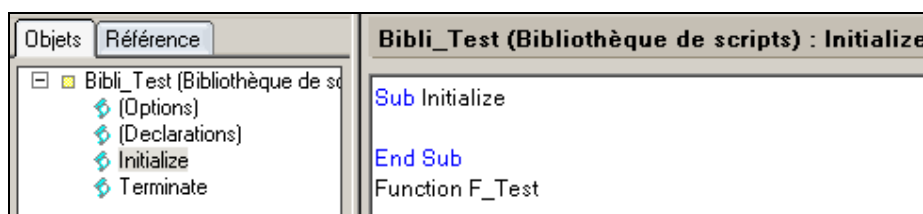
Les bibliothèques de scripts permettent de regrouper des fonctions et procédures pour qu'elles soient accessibles depuis tous les éléments de structure. A partir de la version 6 de Lotus Notes, vous pouvez aussi créer des bibliothèques de Java et de Java Script.

Toute variable déclarée dans le module : « (déclarations) » sera une variable globale, accessible et modifiable depuis toutes les fonctions et procédures de la bibliothèque.

Différence entre fonction et procédure : la fonction renvoie un résultat la procédure non.

Pour ajouter une fonction ou une procédure, il suffit de se placer en dessous du « End Function » ou du « En Sub » et de saisir la ligne de déclaration de la fonction ou de la procédure

```
End Sub
Function F_Test (t1 as string) as Integer
```



11.1 CREER UNE FONCTION

Le nom de la fonction doit être écrit en un seul mot. Le nom de la fonction est automatiquement considéré comme une variable, cette variable est utilisée pour renvoyer l'information elle possède le même « type » que celui déclaré comme variable de renvois.

La liste des variables passées en paramètres n'est pas obligatoire, mais toutes les variables passées en paramètres doivent être renseignées. Toutes les variables passées en paramètres doivent être séparées par une virgule.

Function « Nom_de_la_fonction » (« Nom_Variable_1 » as «type variable », « nom_variable_2 » as «type variable »,) as «type variable à renvoyer»

```
Function MaFonction_Test ( sT1 As String, iN2 As Integer) As String

    MaFonction_Test = Left(sT1,iN2)
    MaFonction_Test = Ucase( MaFonction_Test)

End Function
```

```
Function MonAutreFonction_Test As String

    MonAutreFonction_Test = Left("azerty",3)
    MonAutreFonction_Test = Ucase( MonAutreFonction_Test )

End Function
```

INITIATION AU LOTUS SCRIPT

Utilisation des fonctions déclarées ci-dessus :

```
Dim sText As String
Dim sResultat As String

sText = "qwerty"

sResultat = MaFonction_Test(sText,3)

Msgbox sResultat

sResultat = MonAutreFonction_Test

Msgbox sResultat
```

Pour sortir d'une fonction avant la fin de la fonction, il faut utiliser : « Exit Function »

```
Function MaFonction_Test ( sT1 As String, iN2 As Integer) As String

    If Len(sT1)<iN2 Then
        MaFonction_Test = sT1
        Exit Function
    End If

    MaFonction_Test = Left(sT1,iN2)
    MaFonction_Test = Ucase( MaFonction_Test)

End Function
```

Une fonction peut être utilisée comme une procédure, c'est-à-dire ne pas renvoyer de résultat. Dans ce cas, il faut la faire précéder d'un « Call ».

```
Call MaFonction_Test(Text,3)
```

11.2 CREER UNE PROCEDURE

Une procédure ou « Sub » fonctionne exactement de la même manière qu'une fonction à la seule différence qu'elle ne renvoie pas de résultat. Il faut simplement remplacer le mot « Function » par le mot « Sub ».

Par contre comme une procédure ne renvoi pas de variable la déclaration est plus courte, on ne déclare pas le type de variable à renvoyer.

INITIATION AU LOTUS SCRIPT

Pour appeler la procédure il faut la faire précéder d'un « Call »

Sub « Nom_de_la_procedure » (« Nom_Variable_1 » as «type variable », « nom_variable_2 » as «type variable »)

```
Sub MaProcedure_Test (sT1 As String )
    MsgBox Ucase(sT1)
End Sub

Sub MonAutreProcedure_Test
    MsgBox Ucase("abcdegh")
End Sub
```

Utilisation des procédures déclarées ci-dessus :

```
Dim sResultat As String
Dim sText As String

sText = "azerty"

Call MaProcedure_Test (sText)

Call MonAutreProcedure_Test
```

Pour sortir d'une procédure avant la fin de la procédure il faut utiliser : « Exit Sub »

```
Sub MaProcedure_Test (sT1 As String, iN2 As Integer )
    Dim sT2 As String
    If Len(T1)<iN2 Then
        Exit Sub
    End If

    sT2 = Left(T1,iN2)
    sT2 = Ucase(sT2)

    MsgBox sT2

End Sub
```

11.3 OPTION PUBLIC

Utilisable dans tous les éléments de structures, « Option Public » est d'un grand intérêt dans les bibliothèques car cette option indique si les variables globales, fonctions et procédures sont accessibles (peuvent être appelées) depuis l'extérieur de la bibliothèque (depuis un masque par exemple).

```
Option Public
Option Declare
```

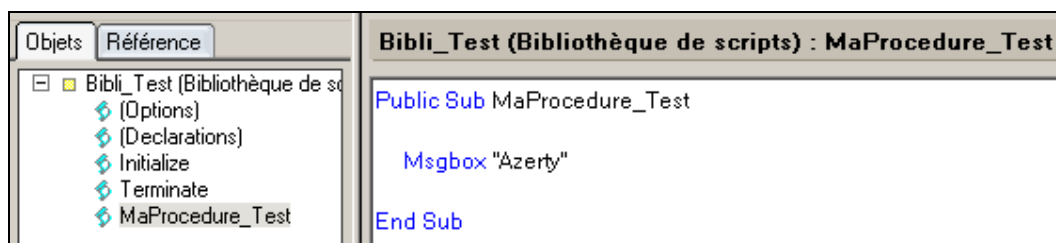
Si dans le module « Option » de la bibliothèque vous ajoutez « Option Public » après (ou avant) « Option Declare », toutes les variables globales, fonctions et procédure sont accessible depuis l'extérieur.

Si vous n'ajoutez pas « Option Public », aucune variable globale n'est accessible depuis l'extérieur. Les fonctions et procédures ne seront accessibles depuis l'extérieur que si vous les faites précéder d'un « Public ».

```
Public Sub MaProcedure_Test

    MsgBox "Azerty"

End Sub
```



Dans le même esprit, si vous faites précéder une fonction ou une variable globale de votre bibliothèque d'un « Private », la fonction n'est utilisable qu'à l'intérieur de la bibliothèque. Cela n'a de sens que si vous avez déclaré votre bibliothèque comme publique « Option Public ».

```
Private Sub MaProcedure_Test

    MsgBox "Qwerty"

End Sub
```

ATTENTION

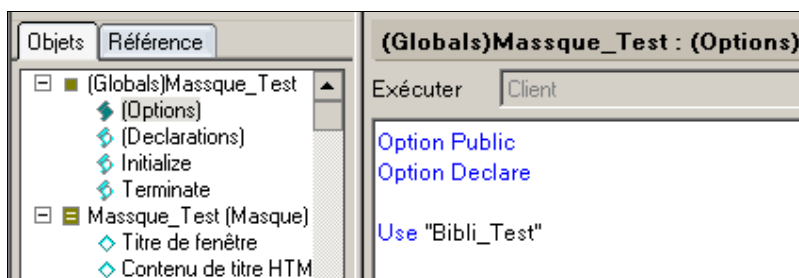
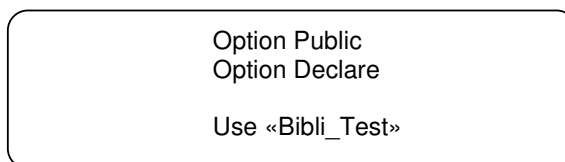
Les variables globales publiques posent un léger problème. Si vous essayez de faire appel à une bibliothèque de script pour utiliser une fonction et que cette bibliothèque possède une variable globale publique, vous ne pourrez pas re-déclarer une variable du même nom dans votre code. Lotus Notes générera immédiatement un message d'erreur dans le designer vous indiquant que cette variable est déjà déclarée.

Vous avez le choix entre changer le nom de votre variable dans votre code, ou bien passer votre variable globale dans la bibliothèque de script en « Private ».

11.4 UTILISATION D'UNE BIBLIOTHEQUE DE SRIPT

Dans la partie « option » de votre élément de design après « option declare », il faut ajouter une ligne : Use « Nom_de_la_bibliotheque », le nom de la bibliothèque doit être entre guillemets. Vous pouvez ensuite utiliser les fonctions et procédures de votre bibliothèque.

Vous pouvez appeler une bibliothèque depuis n'importe quel élément de structure même une autre bibliothèque.



INFORMATION

Vous pouvez créer des fonctions et des procédures dans divers éléments de la structure (Masque, Vue, Bouton, Action ...). Les principes de création et d'utilisation sont identiques, mais les procédures et fonctions ne seront connues et accessibles que dans les éléments de structures dans lesquels elle ont été créés.

Pour créer une fonction ou une procédure, placez-vous juste sous le « end sub » de « l'initialize » ou du « terminate » de l'élément de design et utilisez la même méthode que pour la bibliothèque de script

11.5 LE CHARGEMENT D'UN BIBLIOTHEQUE DE SCRIPT

Il faut savoir qu'une bibliothèque de script est chargée à l'ouverture d'un élément de design (masque, vue). Si après avoir ouvert l'élément de design vous modifiez la bibliothèque (surtout les noms des procédures ou fonctions), vous risquez un message d'erreur du genre « Not a sub or function name ». En effet comme la bibliothèque de script n'a pas été rechargée, la modification n'est pas prise en compte et donc l'appel à la fonction impossible.

Dans ce cas, le mieux, est de mettre en « commentaire » la ligne de code qui pose problème. Ensuite, enregistrer et fermer l'élément de design sur lequel l'on travaille et faite de même avec bibliothèque de script qui pose problème (si vous ne l'avez pas déjà fait). Puis de rouvrir l'élément de design, vous serez peut-être obligé de fermer tout vos clients Notes puis de les rouvrir.

Ne vous en faite pas on se fait tous avoir au moins une fois, on cherche pendant cinq minutes pourquoi ça ne marche pas (alors que ça devrais) avant de se rappeler qu'il faut recharger la bibliothèque de script.

12 DES PRECISIONS SUR DIVERS ASPECTS DU LOTUS SCRIPT

12.1 DECLARATION RAPIDE D'UN OBJET

Comme vous l'avez sans doute remarqué, dans les exemples fournis, la déclaration d'un objet est toujours séparée de son initialisation. Cependant, certains objets permettent de faire ces deux opérations en une seule ligne.

```
Dim Session As NotesSession
Set Session = New NotesSession
```

'Devient

Dim Session As New NotesSession

```
Dim UIWork As NotesUIWorkSpace
Dim UIDoc As NotesUIDocument
```

'Devient

Dim UIWork As New NotesUIWorkSpace

ATTENTION

Il y a cependant une limitation, vous ne pourrez pas utiliser ce type de déclaration dans les modules « (Déclarations) » de vos éléments de design. Le designer de Lotus Notes le refuse catégoriquement.

12.2 FRACTIONNER UNE INSTRUCTION SUR PLUSIEURS LIGNES

Lorsque qu'une instruction de votre code est trop longue, il peut être utile de faire un peu de « mise en page » et d'éclater une ligne en plusieurs lignes pour une meilleure lisibilité du code. Cela ne change absolument rien à son fonctionnement. Elle sera toujours considérée comme une seule ligne par Lotus Notes, mais elle est simplement plus facile à lire.

Il suffit de terminer la ligne par un « under score » « _ » et de commencer la suivante par un « plus » « + ». Il y a quelques limitations, si Lotus Notes n'accepte pas cette manière de saisir le code, entourez la dernière instruction de la ligne par des parenthèses, généralement cela s'avère utile.

```
Dim doc As NotesDocument
Dim sObjet As String
sObjet = Doc.Subject(0)
```

'Cette ligne

```
Msgbox "Mail de "+doc.Sendo(0)+" "+Doc.CopyTo(0)+" "+Doc.blindCopyTo(0)+" avec
comme sujet "+sObjet+"(Le corps du mail n'est pas affiché)",3+32+256," AFFICHAGE DES
INFORMATONS DU MAIL"
```

'Devient

```
Msgbox "Mail de "+doc.Sendo(0)_
+"," +Doc.CopyTo(0)_
+"," +Doc.blindCopyTo(0)_
+" avec comme sujet "+(sObjet)_
+"Le corps du mail n'est pas affiché",3_
+32_
+256." AFFICHAGE DES INFORMATONS DU MAIL "
```

INITIATION AU LOTUS SCRIPT

12.3 LA COMMANDE « FORALL »

La commande « ForAll » utiliser dans plusieurs exemples, permet de passer en revue tous les éléments d'un tableau, d'une liste ou d'un objet dont une propriété renvoie une liste valeur comme NotesItem.Values.

ForAll « variable contenant l'élément traité par le ForAll » In « Liste d'éléments »

```
Dim Doc As NotesDocument
Dim item As NotesItem
Dim lstValue List As String

lstValue(0) = "aaa"
lstValue(1) = "bbb"
lstValue(2) = "ccc"
lstValue(3) = "ddd"

Doc.Test = lstValue
Set Item = Doc.GetFirstItem("Test")

Forall sValeur In item.Values

    MsgBox sValeur

End Forall
```

La variable n'a pas besoin d'être déclarée (même si vous utilisez « Option Declare »), le « ForAll » s'en charge pour vous. L'inconvénient c'est qu'il risque de la déclarer comme un « variant ». Il est donc préférable de déclarer la variable avant (comme une variable normale) lorsque cela est possible, car pour certains types de variables le « Forall » empêche la « pré-déclaration » (type string par exemple).

12.4 GESTION DES ERREURS

Il arrive qu'une instruction de votre code rencontre un problème et génère une erreur, dans ce cas Lotus Notes affiche une boîte de dialogue pas très compréhensible. Le Lotus Script permet une gestion des erreurs, c'est-à-dire de les intercepter pour les gérer.

```
On Error Goto ErrorHandler ' Si erreur aller à l'étiquette

'
' Votre code
'

Exit Sub
ErrorHandler:
MsgBox "Erreur N°: " + Cstr(Err)_ ' code numérique de l'erreur
+"Description : " + Error(Err)_ ' La description de l'erreur
+"Ligne N°: " + Cstr(Erl)_ ' La ligne où se trouve l'erreur
+""",16, " ERREUR !"
Exit Sub
```

INITIATION AU LOTUS SCRIPT

L'exemple ci-dessus arrête la procédure pour afficher un message d'erreur plus compréhensible. Mais si vous désirez tout simplement que Lotus Notes passe à l'instruction suivante sans afficher de message d'erreur :

```
On Error Resume Next
```

Vous pouvez aussi placer le « Resume Next » juste après l'affichage de votre message d'erreur pour que le programme poursuive son traitement sans s'arrêter.

```
On Error Goto ErrorHandler ' Si erreur aller à l'étiquette
,
' Votre code
,
Exit Sub
ErrorHandler:
MessageBox "Erreur N°: " + Cstr(Err)_ ' code numérique de l'erreur
+"Description : " + Error(Err)_ ' La description de l'erreur
+ "Ligne N°: " + Cstr(Erl)_ ' La ligne où se trouve l'erreur
+ "", 16, " ERREUR !"
Resume Next ' Reprend le programme à l'instruction suivante
```

Vous pouvez aussi générer vous-même une erreur

```
On Error Goto ErrorHandler ' Si erreur aller à l'étiquette

On Error Resume Next

Error 9999, "Mon message d'erreur" ' l'erreur portera le numéro 9999

Exit Sub
ErrorHandler:
MessageBox "Erreur N°: " + Cstr(Err)_ ' Numéro du code d'erreur
+"Description : " + Error(Err)_ ' La description de l'erreur
+ "Ligne N°: " + Cstr(Erl)_ ' La ligne où se trouve l'erreur
+ "", 16, " ERREUR !"
Resume Next ' Reprend le programme à l'instruction suivante
```

La commande « Resume Next » permet de ne pas arrêter la procédure et de continuer en reprenant à l'instruction suivante. Elle efface le contenu des variables système d'erreur (Err, Erl et Error), donc pour Lotus Notes il n'y a plus d'erreur !

INITIATION AU LOTUS SCRIPT

12.5 UTILISER LES FORMULES EN LOTUS SCRIPT

Vous allez très vite vous apercevoir que toutes les @formules n'ont pas leurs corollaires en Lotus Script. Vous serez obligé dans certains cas de les re-développer en Lotus Script. Vous pouvez aussi les utiliser en Lotus Script, pour cela utilisez la commande « Evaluate ».

12.5.1 RESTRICTIONS

Vous pouvez utiliser presque toutes les @Formules mais pas les @Command ni les @postedCommand. Les @Formules qui ne sont pas utilisables dans un « evaluate » :

```
@DbManager  
@DbName  
@DbTitle  
@DDEExecute  
@DDEInitiate  
@DDEPoke  
@DDETerminate  
@DialogBox  
@PickList  
@Prompt  
@ViewTitle
```

12.5.2 UTILISATION

La commande « Evaluate » contient deux arguments : la formule à évaluer et le document (dorsal) de référence. Le document de référence n'est nécessaire que si vous désirez faire appel à des champs de ce document. Si votre « evaluate » refuse la formule, passez-là d'abord dans une variable de type « String ».

Si la formule que vous voulez évaluer est fautive (une mauvaise syntaxe par exemple), l'evaluate générera une erreur.

```
Dim vrResultat As Variant  
Dim sFormule As String  
  
sFormule = "@date(@now)"  
vrResultat = Evaluate(sFormule)
```

```
Dim Session As NotesSession  
Dim DB As NotesDatabase  
Dim Doc As NotesDocument  
Dim vrResultat As Variant  
  
Set Session = New NotesSession  
Set DB = Session.currentdatabase  
Set Doc = DB.CreateDocument  
Doc.form = "memo"  
Doc.test = "aaa;bbb;ccc;ddd"  
  
vrResultat = Evaluate({@Explode(test;";")},Doc)
```

12.6 PASSER UNE CHAÎNE DE CARACTÈRES

Vous aurez souvent à passer une chaîne de caractères dans une variable ou dans un argument de fonction. Il existe plusieurs méthodes :

- Encadrer votre texte par des guillemets : "Ma chaîne de caractères"
- Encadrer votre texte par des pipes (AltGr 6) : |Ma chaîne de caractères|
- Encadrer votre texte par des accolades : {Ma chaîne de caractères}

Le symbole que vous utilisez pour déclarer une chaîne de caractères ne peut pas être directement utilisable dans votre chaîne comme un caractère, pour cela il doit être doublé.

```
Msgbox "Valeur du champ : ""AAAA"" "
```

```
Msgbox |Valeur du champ : ||BBBB|| |
```

Mais si le symbole n'est pas utilisé pour déclarer une chaîne de caractères vous pouvez l'utiliser normalement

```
Msgbox {Valeur du champ : "AAAA"}
```

12.7 LES COMMENTAIRES

Il existe deux manières de mettre des commentaires en Lotus Script.

Pour mettre un bloc d'instructions ou de texte en commentaire il suffit de l'encadrer entre un « %Rem » et un « %End Rem ». Tout ce qui se trouvera entre ces deux balises sera considéré comme du commentaire. Peu importe le nombre de lignes.

```
Sub Initialize|  
  
Dim Session As NotesSession  
Dim DB As NotesDatabase  
Dim Doc As Notesdocument  
  
Set Session = New NotesSession  
Set DB = Session.currentdatabase  
Set Doc = DB.CreateDocument  
  
%REM  
Mise en commentaire  
DOC.FORM = "MEMO"  
Call Doc.save(true,false)  
Call doc.remove(true)  
%END REM  
  
End Sub
```

```
Dim Session As NotesSession  
Dim DB As NotesDatabase  
Dim Doc As Notesdocument  
  
Set Session = New NotesSession  
Set DB = Session.currentdatabase  
Set Doc = DB.CreateDocument  
  
%REM  
Mise en commentaire  
DOC.FORM = "MEMO"  
Call Doc.save(true,false)  
Call doc.remove(true)  
%END REM
```

INITIATION AU LOTUS SCRIPT

Pour mettre juste une ligne en commentaire ou une partie d'une ligne, il suffit de placer une coche (') devant la partie de la ligne à mettre en commentaire.

```
Dim Session As NotesSession
Dim DB As NotesDatabase
Dim Doc As Notesdocument

Set Session = New NotesSession
Set DB = Session.currentdatabase
Set Doc = DB.CreateDocument 'Creation du document
Doc.form = "memo"
Call Doc.save(True,False)
' Call doc.remove(True)
```

```
Sub Initialize

    Dim Session As NotesSession
    Dim DB As NotesDatabase
    Dim Doc As Notesdocument

    Set Session = New NotesSession
    Set DB = Session.currentdatabase
    Set Doc = DB.CreateDocument 'Creation du document
    Doc.form = "memo"
    Call Doc.save(True,False)

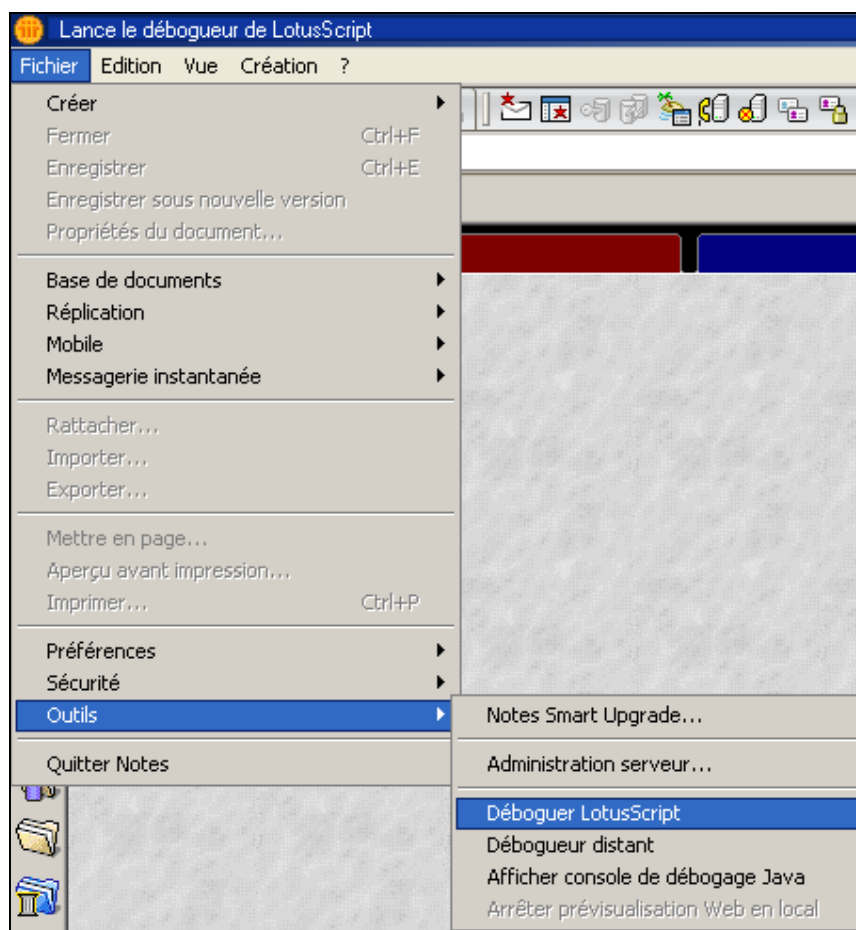
    'Call doc.remove(True)

End Sub
```

13 L'UTILISATION DU « DEBOGUEUR LOTUS SCRIPT »

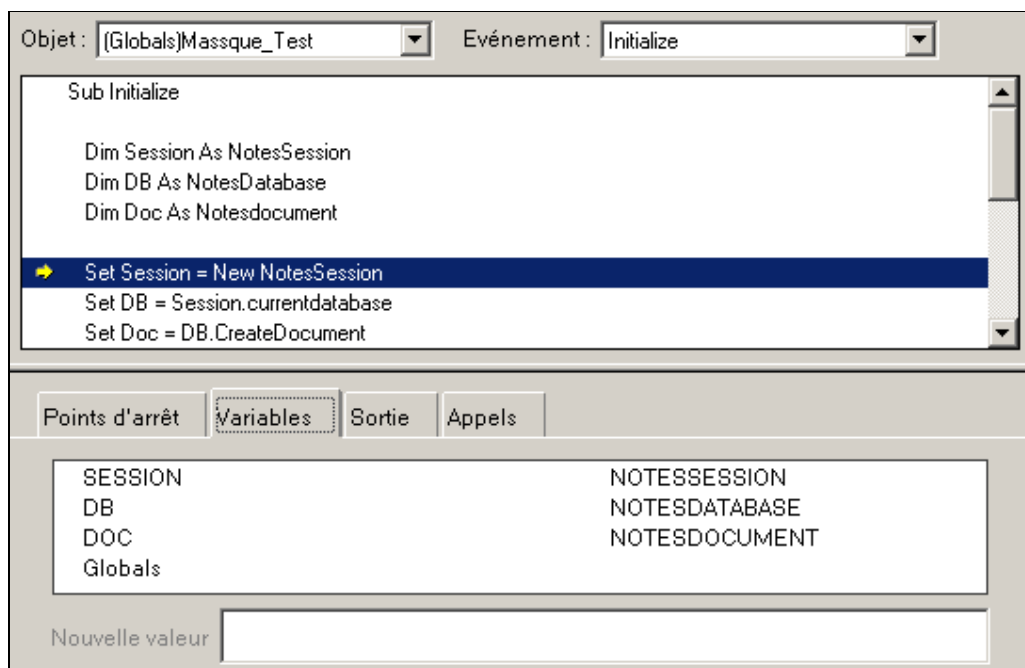
Le débogueur est l'outil indispensable pour faire du développement en Lotus Script. Sans lui, la recherche des anomalies et autres bugs serait plus de l'ordre de la divination que de la programmation. Il permet de voir le déroulement du code instruction par instruction et l'état des variables. Il vous permettra entre autres de vérifier que vos différentes variables et objets sont bien renseignés et/ou initialisés, de voir comment tournent vos différentes boucles ou si les conditions de vos « if » sont correctes.

Pour activer le débogueur, menu : Fichier\Outils\Débogueur LotusScript
 Pour le désactiver, c'est la même procédure.



Le débogueur est composé de deux parties, la première affiche les instructions (une petite flèche jaune au début de l'instruction indique où en est le déroulement du script). La deuxième affiche toutes les variables et objets ainsi que leur contenu (qui évolue en temps réel, au fur et à mesure de l'exécution des instructions). Les boutons en haut permettent de gérer le déroulement du script, mode pas à pas pour s'arrêter après chaque instruction, continuer pour que le script finisse de s'exécuter normalement.

INITIATION AU LOTUS SCRIPT

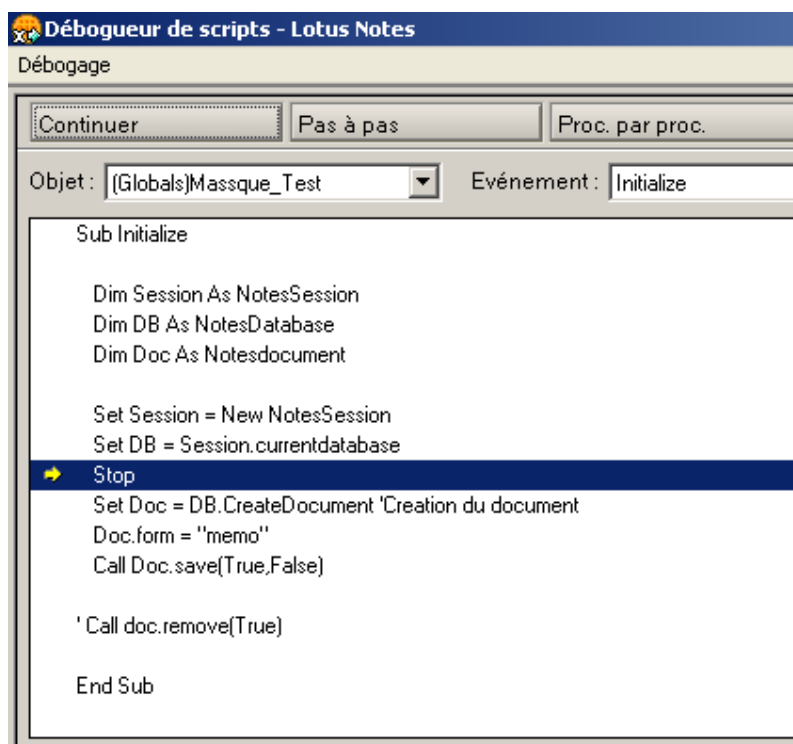


Si le code est particulièrement long, il peut être intéressant de le faire se dérouler normalement puis de l'arrêter à l'endroit qui pose problème (ou du moins qu'il semble poser problème). Pour cela, il y a deux possibilités : le « stop » et le « point d'arrêt ».

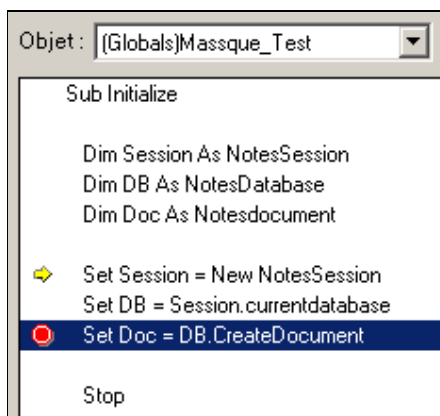
Le « Stop » est une commande Lotus script qui ne fonctionne que lorsque le débogueur est activé et qui stoppe le script pour pouvoir passer en mode pas à pas (instruction après instruction).

```
Dim Session As NotesSession  
Dim DB As NotesDatabase  
Dim Doc As Notesdocument  
  
Set Session = New NotesSession  
Set DB = Session.currentdatabase  
Set Doc = DB.CreateDocument  
  
Stop  
  
Doc.form = "memo"  
Call Doc.save(True,False)  
  
Stop  
  
Call doc.remove(True)
```

INITIATION AU LOTUS SCRIPT



Le « Point d'arrêt » fait exactement la même chose mais il est positionné via le débogueur lui-même. Pour placer un « Point d'arrêt », il suffit de se positionner sur la ligne avant laquelle on veut s'arrêter et de faire un double clic, un petit point rouge apparaît. Pour le retirer, il suffit de faire un autre double clic sur la même ligne.



L'avantage avec les « Stop » et les « Point d'arrêt » c'est qu'ils n'ont strictement aucune influence tant que le débogueur n'est pas activé !

CONCLUSION (IL EN FAUT BIEN UNE)

Ce tutorial n'aura pas fait de vous un expert en Lotus Script, mais il vous aura donné les notions de bases pour commencer sereinement et faire les traitements les plus courants. Les fonctions les plus importantes vous ont été expliquées afin de répondre aux besoins et questions les plus récurrentes. L'acquisition de ces connaissances, leur utilisation puis leur approfondissement se feront par la pratique, elle est votre meilleur allié (avec l'aide en ligne du designer). C'est en faisant des erreurs que l'on progresse, l'expérience personnelle que vous en retirez, vous sera plus que bénéfique. Mais n'oubliez pas que la connaissance des objets, méthodes et procédures n'est pas une fin en soi, c'est leur utilisation appropriée dans un algorithme bien réfléchi qui fait souvent toute la différence.

Maintenant c'est à vous de jouer... enfin de coder !