La gestion des erreurs dans Excel

par SilkyRoad (silkyroad.developpez.com)

Date de publication : 25/02/2007

Dernière mise à jour : 01/04/2007

Ce document décrit les types d'erreurs pouvant survenir dans Excel (formules et macros) ainsi que les outils disponibles pour gérer ces erreurs.

Tous les exemples proposés ont été testés avec Excel2002.

I - Introduction

Des erreurs peuvent parfois apparaître dans les formules de calcul ou lors de l'exécution d'une macro.

L'erreur peut être la conséquence d'une saisie erronée, mais peut parfois aussi être inévitable: Exemple des formules dans un tableau de bord prérempli et complété à chaque fin de mois. Les cellules contenant des formules pour les prochains mois peuvent logiquement renvoyer une erreur car les données ne sont pas encore renseignées. De la même façon, une macro qui vérifie l'existence d'une feuille peut logiquement renvoyer une erreur si l'onglet n'existe pas.

Excel dispose de plusieurs outils qui permettent d'identifier et gérer ces erreurs. Il est intéressant de connaître la signification les codes d'erreur, de cerner la cause et pouvoir ainsi trouver une solution corrective plus facilement. C'est l'objet de ce tutoriel.

II - Les formules

II-A - Les types d'erreurs

Un petit triangle vert, dans l'angle supérieur gauche, permet de visualiser rapidement les cellules contenant des erreurs.

Les formules Excel renvoient des valeurs d'erreur spécifiques en fonction du problème rencontré:

#NUL! :

Survient lorsque vous spécifiez une intersection de deux zones qui, en réalité, ne se coupent pas.

* Par exemple lors de l'utilisation d'un opérateur de plage incorrect (=SOMME(A1 A10)). Il manque les deux-points (:) dans la formule pour séparer la référence de la première cellule de la référence de la dernière cellule.

#DIV/0! :

Survient lorsqu'un nombre est divisé par zéro.

#VALEUR! :

Survient lorsqu'un argument ou un élément de la formule est inapproprié.

* Vous avez attribué une plage à un opérateur ou à une fonction qui exige une seule valeur et non pas une plage.

* Les éléments de la formule ne sont pas compatibles (Par exemple =10+"mimi").

* Il s'agit d'une formule matricielle qui doit être revalidée: Dans ce cas sélectionnez la cellule, touche F2 puis appuyez sur CTRL+MAJ+ENTRÉE.

* Les dimensions de la matrice sont incorrectes.

#REF! :

Survient lorsque les coordonnées d'une cellule ne sont pas valides.

* Lors de l'utilisation d'une liaison non valide (Vérifiez le format de la liaison ='C:\dossier\[NomClasseur.xls]NomFeuille'!A1).

* Lorsque la liaison vers une rubrique d'échange dynamique de données (DDE ou Dynamic Data Exchange) n'est pas disponible.

* Après la suppression ou le collage de cellules auxquelles d'autres formules font référence.

#NOM? :

Survient lorsque l'application ne reconnaît pas le texte dans une formule.

* Vérifiez l'existence et l'orthographe des cellules et plages nommées.

* Vérifiez l'existence et l'orthographe des fonctions utilisées.

* Vérifiez la présence deux points (:) nécessaires pour référencer une plage de cellules.

* Vérifiez que l'utilisation des étiquettes est bien autorisée:

Menu Outils/Options/Sélectionnez l'onglet "Calcul"/Cochez l'option "Accepter les étiquettes dans les formules."/Cliquez sur le bouton OK pour valider.

* Vérifiez que les textes sont encadrés par des guillemets (Par exemple =RECHERCHE("mimi";A:A)).

* Si la formule fait référence à des valeurs ou à des cellules d'autres feuilles de calcul ou d'autres classeurs dont le nom contient un caractère non alphabétique ou un espace,

vérifiez que vous avez bien placé une apostrophe (') de part et d'autre du nom (='Nom Feuille'!C9).

#NOMBRE! :

Survient si formule ou une fonction contient des valeurs numériques non valides.

* Lorsqu'un nombre est trop grand ou trop petit pour être représenté dans Excel. Les valeurs doivent être compris entre $-1*10^{307}$ et $1*10^{307}$.

* Lorsqu'une fonction qui s'exécute par itération ne parvient pas à trouver un résultat.

Pour résoudre le problème, dans le menu Outils/Options/onglet "Calcul"/Cochez l'option "Itération".

* Lorsqu'un argument est incorrect dans une fonction qui exige un argument numérique.

#N/A :

Survient lorsqu'une valeur nécessaire au bon fonctionnement de la formule est manquante.

* Si la dimension des plages de cellules n'est pas identique dans les formule matricielles:

Par exemple, =SOMMEPROD((A1:A10="dvp")*(B1:B9="number one")) renvoie une erreur. Vous devez écrire: =SOMMEPROD((A1:A10="dvp")*(B1:B10="number one"))

* Si des cellules référencées dans la formule contiennent des valeurs #N/A ou NA().

* Si une fonction personnalisée n'est pas disponible.

* Si un argument obligatoire est absent, ou d'un type inapproprié dans la fonction.

* Si les fonctions RECHERCHEV, RECHERCHEH ou INDEX effectuent une recherche dans une ligne ou une colonne non triée.

Spécifiez la valeur FAUX dans le dernier argument de ces fonctions. Elles peuvent ainsi effectuer une recherche dans une ligne ou une colonne non triée.

:

* Survient lorsqu'une colonne n'est pas suffisamment large pour afficher la totalité d'une donnée numérique:

Modifiez tout simplement la largeur de la colonne afin de régler le problème.

* Survient lorsque les calculs sur les dates et les heures donnent des résultats négatifs.

Si vous utilisez le calendrier depuis 1900, les dates et les heures doivent impérativement être positives.

1ere solution. Modifiez le format de la cellule contenant la formule:

Clic droit dans la cellule/Format de cellule/ Onglet "Nombre"/Sélectionnez la catégorie "Standard" par exemple/Cliquez sur le bouton OK pour valider.

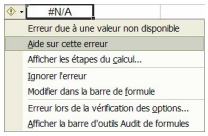
2eme solution. Passez en calendrier depuis 1904: Menu outils/Options/Onglet "Calcul"/Cochez l'option "Calendrier depuis 1904".

II-B - L'aide à la résolution des erreurs

Vous pouvez utiliser l'aide Excel (F1) pour obtenir des informations très détaillées sur chaque type d'erreur.

Depuis Excel2002, une balise active apparaît lorsque vous sélectionnez une cellule qui contient une erreur.

Cliquez sur ce bouton pour afficher un menu d'aide à la résolution des problèmes.



Les options du menu sont adaptées au type d'erreur:

* Description du type d'erreur.

* Afficher l'aide Excel associée à l'erreur identifiée.

* Modifier la formule.

* Evaluer la formule.

* Masquer la balise et le petit triangle vert.

* Repérer les antécédents contenant des erreurs.

* Afficher la barre d'outils d'Audit des formules.

* Afficher la boîte de dialogue d'options pour la vérification des erreurs.

La barre d'outils d'**Audit des formules** propose aussi plusieurs outils pour identifier les erreurs.

(Menu Outils/Audit de formules/Afficher la barre d'outils)

Sélectionnez la cellule contenant la formule puis cliquez sur le bouton "**Repérer les antécédents**".

La commande dessine des flèches d'audit à partir des cellules qui fournissent directement des valeurs à la formule active (antécédents).

Une flèche rouge signifie que la cellule antécédente contient une erreur.

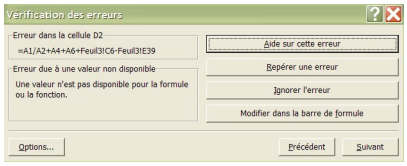
Pour repérer les formules qui fournissent indirectement les valeurs à la formule de la cellule active, cliquez à nouveau sur le bouton "**Repérer les antécédents**".

Cette commande dessine une flèche d'audit vers la cellule active, à partir des cellules spécifiées dans la formule, si cette dernière renvoie une erreur.

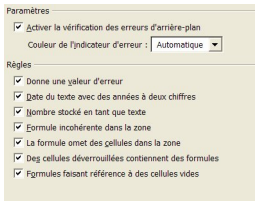
Evalue la formule étape par étape.

Identifie toutes les formules qui contiennent des valeurs non comprises dans les limites définies par le menu Données/Validation.

Le bouton "Vérification des erreurs" reprend les options d'aide sous forme d'une boîte de dialogue.



Les paramètres et les règles de vérification d'erreurs sont accessibles depuis le menu Outils/Options/Onglet "Vérification des erreurs".



II-C - Les formules pour gérer les erreurs

Excel dispose aussi de fonctions spécifiques pour identifier et gérer les erreurs.

ESTERR :

Renvoie VRAI si la cellule contient une erreur (autre que #NA).

=ESTERR(A1)

ESTERREUR :

Renvoie VRAI si la cellule contient une erreur (#NA compris).

=ESTERREUR(A1)

La fonction **ESTERREUR**, associée à une condition **SI**, permet de masquer les messages erreurs.

Cette formule affiche "" dans la cellule si la division A1/A2 renvoie une erreur:

=SI(ESTERREUR(A1/A2);"";A1/A2)

Un exemple pour additionner les valeurs d'une plage, dont certaines cellules contient des erreurs.

{=SOMME(SI(ESTERREUR(A1:A5);"";A1:A5))}

Formule à valider par Ctrl+Maj+Entree.

Utilisez les formats conditionnels pour masquer toutes les valeurs d'erreur dans une feuille:

Sélectionnez toutes les cellules, puis le Menu Format/Mise en forme conditionnelle.

Dans le champ "La formule est:", saisissez =ESTERREUR(A1)

Puis Format/Police/Sélectionnez la police blanche.

Cliquez sur le bouton OK dans les différentes boîtes de dialogue.

Les valeurs d'erreur sont désormais toutes masquées.

ESTNA :

Vérifie si la cellule contient une erreur type #N/A (Renvoie VRAI ou FAUX).

=ESTNA(A1)

TYPE.ERREUR :

Renvoie une valeur en fonction du type d'erreur dans la cellule.

=TYPE.ERREUR(A1)

#NUL!: renvoie 1

#DIV/0!: renvoie 2

#VALEUR!: renvoie 3

#REF!: renvoie 4

#NOM?: renvoie 5

#NOMBRE!: renvoie 6

#N/A: renvoie 7

La fonction renvoie #N/A si la cellule ne contient pas d'erreur.

NA :

Renvoie la valeur d'erreur #N/A dans la cellule contenant cette formule.

=NA()

La fonction NA permet de marquer les cellules vides et évite d'inclure involontairement des cellules vides dans les calculs.

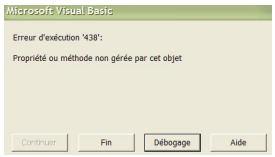
Vous pouvez aussi saisir directement #N/A dans la cellule.

III - Les macros

III-A - Comment gérer les erreurs

Les erreurs d'exécution se produisent lorsqu'une instruction tente d'accomplir une opération non valide.

Chaque erreur d'exécution provoque l'arrêt des macros. La ligne incriminée est surlignée en jaune et un message d'alerte s'affiche à l'écran.



Le numéro d'erreur ainsi qu'une description courte s'affichent dans la boîte de dialogue.

Le bouton **Fin** permet de sortir immédiatement de la procédure.

Le bouton **Débogage** ferme le message d'alerte et affiche la ligne qui a déclenché l'erreur.

Le bouton **Aide** ouvre l'aide associée au code d'erreur.

L'application dispose d'un gestionnaire d'erreurs pour prendre en compte ce genre de désagréments.

C'est l'instruction **On Error** qui valide la gestion d'erreur. Le gestionnaire est ensuite activé lorsque survient une erreur.

III-A-1 - On Error GoTo

Effectue un branchement vers une ligne spécifiée. Par exemple On Error goTo errorHandler

Cette instruction indique l'emplacement de la procédure qui gère les erreurs. Lorsqu'une erreur survient, l'exécution de la macro se déplace automatiquement jusqu'à cette ligne: errorHandler, dans l'exemple ci dessous.

On Error GoTo errorHandler doit être placé juste après la déclaration des variables, ou au moins avant la première ligne de procédure risquant de provoquer une erreur.

errorHandler: doit être placé en fin de procédure, éventuellement précédé par l'instruction Exit Sub.

Vba

```
Sub laMacro()
    'Placé en début de macro:
```

Vba
```
        'Si une erreur survient, on va à la ligne "errorHandler"
        On Error GoTo errorHandler

        '
        'La procédure
        '

        'Permet de sortir de la procédure et évite la gestion d'erreur (errorHandler), si la macro
        's'est déroulée sans encombre.
        Exit Sub

    errorHandler:
        'indique le numéro et la description de l'erreur survenue
        MsgBox Err.Number & vbLf & Err.Description
    End Sub
```

S'il n'y a pas de problème, la macro suit son cours et se termine par l'instruction Exit Sub. Dans le cas contraire, la macro va directement à la ligne errorHandler: pour gérer l'erreur.

Vous pouvez aussi ajouter l'instruction **Resume** si vous souhaitez reprendre l'exécution au niveau de la ligne à l'origine de l'erreur:

Vba
```
    Sub UtilisationInstructionResume()
        Dim i As Integer, j As Integer

        On Error GoTo ErrorHandler

        'Par défaut i=0 (et va provoquer une erreur)
        j = 5 / i
        MsgBox j


        Exit Sub

    ErrorHandler:
        MsgBox "Erreur: " & Err.Number & vbCrLf & Err.Description
        If i = 0 Then i = 1
        'Reprend l'exécution au niveau de la ligne à l'origine de l'erreur.
        Resume

    End Sub
```

III-A-2 - On Error GoTo 0

Invalide le gestionnaire d'erreurs précédemment créé par l'instruction On Error Goto.

Chaque erreur d'exécution provoque l'arrêt des macros, après la ligne **On Error GoTo 0**.

Dans l'exemple suivant, le gestionnaire d'erreur joue son rôle jusqu'à la ligne **On Error GoTo 0**. Ensuite la division par zéro fait planter la procédure.

Vba

```
Sub laMacro()
    Dim x As Double

    'Permet d'atteindre la ligne "errorHandler" si une erreur survient.
    On Error GoTo errorHandler

    MsgBox 5 / 2 'Pas de problème
    x = 5 / 0 'Division par zero = erreur
    MsgBox x

errorHandler:

    'Invalide le gestionnaire d'erreur.
    On Error GoTo 0

    'Provoque une erreur d'exécution car il n'y a plus de gestion.
    MsgBox 5 / 0

End Sub
```

III-A-3 - On Error Resume Next

Permet de continuer la procédure et passe directement à la ligne suivante, en cas d'erreur.

Cette instruction est pratique pour contourner une ligne pouvant poser problème.

Par exemple, lorsque vous vérifiez l'existence d'une feuille dans le classeur:

Vba

```
Sub laMacro()
    Dim Ws As Worksheet

    'Evite le message d'erreur si la feuille n'existe pas.
    On Error Resume Next
    Set Ws = ThisWorkbook.Worksheets("NomFeuille")
    On Error GoTo 0

    '(Ws = Nothing quand l'objet attribué à la variable n'existe pas)
    If Not Ws Is Nothing Then
        MsgBox "La feuille existe dans le classeur."
        Else
        MsgBox "La feuille n'existe pas dans le classeur."
    End If
End Sub
```

Cette instruction doit être employée avec prudence car elle masque tous les bugs qui peuvent survenir dans la procédure. Vous ne recevrez aucune information pour localiser une erreur qui pourrait avoir une influence sur le résultat final de votre macro. Utilisez **On Error Resume Next** seulement si vous ne pouvez pas faire autrement, ou de façon ciblée sur une ligne précise (Comme dans l'exemple précédent).

III-B - Description de l'objet Err

L'objet **Err** contient des informations sur les erreurs d'exécution. Lorsqu'une erreur se produit, les propriétés de l'objet Err stockent d'une part des informations qui identifient l'erreur et d'autre part des informations permettant de gérer cette erreur.

III-B-1 - Clear

La méthode **Clear** réinitialise le contenu de l'objet **Err** à la suite du traitement d'une erreur.

Les propriétés de l'objet **Err** sont alors remises à zéro ou remplacées par des chaînes de longueur nulle.

La méthode Clear est appelée automatiquement dès que l'une des instructions suivantes est exécutée:

* Tout type d'instruction Resume.

* Exit Sub, Exit Function ou Exit Property.

* Toute instruction On Error.

Si la méthode Clear n'était pas utilisée dans l'exemple suivant, le message s'afficherait à chaque tour de boucle après le déclenchement de la première erreur.

Vba

```
Sub laMacro()
    Dim i As Integer
    Dim Cible As Byte

    On Error Resume Next

    For i = 1 To 5
        'Renvoie une valeur aléatoire entre 1 et 400
        Randomize
        Cible = Int((400 * Rnd) + 1)
        'Une erreur survient si le résultat est supérieur à 255
        '(Les variables Byte acceptent uniquement des valeurs entre 0 et 255)

        If Err.Number <> 0 Then
            MsgBox "Erreur"
            'Réinitialise les propriétés de l'objet Err.
            Err.Clear
        End If

    Next i
End Sub
```

III-B-2 - Description

La propriété **Description** renvoie ou définit un chaîne de caractères qui correspond à une courte description de l'erreur.

Vba

```
Sub DescriptionErreur()
    Dim Obj As OLEObject

    On Error Resume Next

    'Déclenchement erreur:
    'Spécifie un objet qui n'existe pas dans la feuille
    Set Obj = Worksheets("Feuil1").Textbox1

    MsgBox Err.Description
End Sub
```

III-B-3 - HelpFile

La propriété **HelpFile** renvoie ou définit le chemin d'accès complet à un fichier d'aide.

Vba

```
Sub FichierAideErreurs()
    Dim x As Double

    On Error Resume Next
    'La division par zéro va provoquer une erreur.
    x = 5 / 0

    MsgBox Err.HelpFile
End Sub
```

III-B-4 - HelpContext

La propriété **HelpContext** renvoie ou définit l'identificateur de contexte associé à une rubrique d'un fichier d'aide.

Cette propriété permet d'afficher automatiquement la rubrique indiquée dans la propriété **HelpFile**. Si les deux propriétés HelpFile et HelpContext sont vides, la valeur de la propriété **Number** est vérifiée. Si la valeur de la propriété Number correspond à la valeur d'une erreur d'exécution Visual Basic, l'identificateur de contexte de l'aide Visual Basic pour cette erreur est utilisé. Si la valeur de la propriété Number ne correspond pas à une erreur Visual Basic, l'écran Sommaire du fichier d'aide Visual Basic s'affiche.

Vba

Vba
```
Dim Msg As String
Dim Cellule As Range

On Error Resume Next
'Va provoquer une erreur: Cells(0, 1) ne peut pas exister
Set Cellule = Cells(0, 1)

MsgBox Err.HelpContext

Msg = "Cliquez sur le bouton AIDE pour consulter la rubrique relative à cette erreur."
MsgBox Msg, vbMsgBoxHelpButton, "Erreur", Err.HelpFile, Err.HelpContext
```

III-B-5 - LastDllError

La propriété **LastDllError** renvoie un code d'erreur système produit lors de l'appel d'une bibliothèque de Liaisons Dynamiques (DLL).

L'exemple suivant, adapté de **l'aide MSDN**, montre comment utiliser la propriété **LastDllError** après avoir appelé une fonction dans l'API Windows. La procédure PrintWindowCoordinates recherche la position d'une fenêtre à partir de son Handle, en appellant la fonction GetWindowRect. Si vous passez un Handle qui n'est pas valide, une erreur se produit, laissant apparaître le numéro d'erreur via la propriété **LastDllError**. La fonction LastDLLErrorDescription permet de retrouver la description associée à la valeur d'erreur.

Vba
```
Option Explicit

Private Const FORMAT_MESSAGE_FROM_SYSTEM = &H1000
Private Const FORMAT_MESSAGE_IGNORE_INSERTS = &H200

Private Declare Function FormatMessage Lib "kernel32" Alias "FormatMessageA" _
    (ByVal dwFlags As Long, lpSource As Any, ByVal dwMessageId As Long, ByVal dwLanguageId As Long,
 _
    ByVal lpBuffer As String, ByVal nSize As Long, Arguments As Long) As Long


'--- Permet de retrouver les coordonnées d'une fenêtre à partir de son Handle
Declare Function GetWindowRect Lib "user32" _
  (ByVal Hwnd As Integer, ByRef lpRect As RECT) As Integer

Public Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type


Private Sub PrintWindowCoordinates(ByVal Hwnd As Integer)
    Dim rectWindow As RECT

  'Une erreur survient si la fonction renvoie 0
  If GetWindowRect(Hwnd, rectWindow) = 0 Then

      'Affiche la valeur d'erreur et la description.
      MsgBox "Valeur d'erreur: " & Err.LastDLLError & vbCrLf & _
        LastDLLErrorDescription(Err.LastDLLError)
```

Vba

```
    Else
        Debug.Print (rectWindow.Bottom)
        Debug.Print (rectWindow.Left)
        Debug.Print (rectWindow.Right)
        Debug.Print (rectWindow.Top)
    End If

End Sub
'---


'Cette fonction permet de retrouver la description associée à la valeur d'erreur
'de la DLL.
'Source:
'http://undim.blogspot.com/2006/12/function-that-microsoft-forgot-to.html
Public Function LastDLLErrorDescription(LastDLLError As Long) As String
    Dim strBuffer As String
    Dim LenBuff As Long
    Dim cRes As Long

    LenBuff = 256
    strBuffer = Space$(LenBuff)

    cRes = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM Or FORMAT_MESSAGE_IGNORE_INSERTS, _
        0&, LastDLLError, 0&, strBuffer, LenBuff, 0&)

    If cRes = 0 Then
    strBuffer = "FormatMessage API execution Error. Couldn't fetch error description."
    Else
    strBuffer = Left$(strBuffer, cRes - 2)
    End If

    LastDLLErrorDescription = strBuffer
End Function



Sub Test()
    PrintWindowCoordinates 1
End Sub
```

III-B-6 - Number

La propriété **Number** renvoie ou définit une valeur numérique spécifique en fonction du type d'erreur. Il s'agit de la propriété par défaut de l'objet **Err**.

Consultez le chapitre **III-D-1** si vous souhaitez lister les codes d'erreurs définis dans Excel.

L'exemple suivant lit le commentaire de la cellule A1. La procédure renvoie une erreur 91 si le commentaire n'existe pas.

Vba

```
Sub lireCommentaire()
    Dim Cm As Comment
```

```vba
        On Error GoTo Fin

        Set Cm = Range("A1").Comment
        MsgBox Cm.Text


        Exit Sub
    Fin:
        If Err.Number = 91 Then MsgBox "Il n'y a pas de commentaire dans la cellule."
    End Sub
```

III-B-7 - Raise

La méthode **Raise** est utilisée pour générer une erreur d'exécution dans la macro.

La syntaxe: Err.Raise Number, Source, Description, HelpFile, HelpContext

Number:

Identifie la nature de l'erreur. La plage de 0 à 512 est réservée aux erreurs système. La plage de 513 à 65535 est disponible pour les erreurs définies par l'utilisateur.

Source:

(Facultatif) Nomme l'objet ou l'application à l'origine de l'erreur.

Description:

(Facultatif) Spécifie la chaîne de caractères qui décrit l'erreur.

HelpFile:

(Facultatif) Définit le chemin d'accès complet au fichier d'aide associé à cette erreur.

Si ce chemin n'est pas indiqué, la procédure adopte le fichier d'aide de Visual Basic.

Consultez le tutoriel de **ThierryAIM** pour **créer vos fichiers d'aide .chm**.

HelpContext:

(Facultatif) Désigne l'identificateur de contexte contenu dans le fichier d'aide.

Cet exemple vérifie le contenu d'un InputBox et crée une gestion d'erreur personnalisée en fonction des données saisies:

Vba

```
Sub Macro_TestRaise()
    Dim Resultat As Variant

    On Error GoTo Fin

    ' Affiche la boîte de dialogue sur la position 100, 100.
    Resultat = InputBox("Saisissez une valeur entre 0 et 9:", "Le titre", 0)
    If Resultat = "" Then Exit Sub

    'Vérifie si la saisie est numérique
    If Not IsNumeric(Resultat) Then _
        Err.Raise 5010, , "Type de donnée non compatible."

    'Vérifie si la saisie est un chiffre en 0 et 9
    If Len(Resultat) > 1 Then _
        Err.Raise 5020, , "La valeur doit être comprise entre 0 et 9", _
        "C:\MonFichierAide.chm", 1000028


    '
    'La procédure
    '
    '


Exit Sub
Fin:

MsgBox "Erreur: " & Err.Number & vbCrLf & _
    Err.Description, vbMsgBoxHelpButton, , Err.HelpFile, Err.HelpContext
End Sub
```

III-B-8 - Source

La propriété **Source** renvoie ou définit une chaîne de caractères indiquant le nom de l'objet ou de l'application qui a généré l'erreur.

III-C - La fonction ERL

La fonction **ERL** permet de récupérer le numéro de ligne qui a provoqué une erreur dans la macro.

Vous devez avoir préalablement numéroté les lignes de procédure dans l'éditeur de macros.

Vba

```
Sub laProcedure()
4     Dim x As Integer

5     x = 0
6     On Error GoTo errHandler

8     x = 5 / 0 'Création erreur (division par zero)


10     Exit Sub
11 errHandler:
12     MsgBox "Une erreur est survenue, Ligne: " & Erl() & _
        vbCrLf & "Numéro d'erreur: " & Err.Number & vbCrLf & Err.Description
14 End Sub
```

Si la ligne x = 5 / 0 n'est pas numérotée, la fonction **ERL** renvoie le numéro de la derniere ligne précédant l'erreur (6 dans l'exemple précédent).

ERL renvoie 0 s'il n'y a pas d'erreur dans la procedure.

ERL renvoie 0 si aucune ligne n'est numérotée dans la macro.

Il existe des outils gratuits, tel que **MZ-Tools**, qui vous aideront à numéroter les lignes de macros très facilement.

III-D - Divers

III-D-1 - Lister les codes d'erreurs dans une feuille de calcul

Les codes d'erreurs récupérables sont stockés dans le fichier d'aide VbLR6.chm (Excel2002).

Cette macro liste dans une feuille de calcul les numéros d'erreur, les descriptions, Les chemins du fichier d'aide et leur index.

Vba

```
Sub ListeErreur()
    Dim Cible As Integer
    Dim I As Integer, j As Integer

    j = 1
    On Error Resume Next

    For I = 1 To 800 'Adaptez le nb de codes erreurs
        Cible = I
        Err.Clear
        Err.Raise Cible

        j = j + 1
        Cells(j, 1) = Err.Number
        Cells(j, 2) = Err.Description
```

```vba
            Cells(j, 3) = Err.HelpFile
            Cells(j, 4) = Err.HelpContext
        Next I

    End Sub
```

Vous pouvez ensuite placer cette deuxième macro dans le **module objet** de la feuille de calcul qui contient la liste les codes d'erreurs précédemment créée.

```vba
    Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Excel.Range, Cancel As Boolean)
        Dim Fso As Object, oFichier As Object
        Dim Ligne As Integer
        Dim Fichier As String

        Ligne = ActiveCell.Row

        Set Fso = CreateObject("Scripting.FileSystemObject")
        Set oFichier = Fso.GetFile(Cells(Ligne, 3))
        Fichier = Fso.GetFileName(oFichier)

        'Affiche l'aide
        Application.Help Fichier, Cells(Ligne, 4)

    End Sub
```

Désormais, la rubrique d'aide s'affiche si vous double cliquez sur une des lignes contenant les codes d'erreur.

III-D-2 - Identifier automatiquement l'apparition des erreurs dans la feuille de calcul

Cette procédure évènementielle Worksheet_Calculate affiche un message d'alerte dès qu'une formule provoque une erreur.

Une option permet d'ouvrir l'aide Excel.

```vba
    Option Explicit

    Private Sub Worksheet_Calculate()
    'Testé sous XP
    Dim Valeur As Range
    Dim Resultat As String, Message As String

    For Each Valeur In Worksheets("Feuil1").UsedRange
        If WorksheetFunction.IsError(Valeur) = True Then
```

Vba
```
            'Affiche les flèches d'audit qui passent par les antécédents
            Valeur.ShowErrors

            Select Case Valeur
            Case CVErr(xlErrDiv0)
            Resultat = "#DIV/0!"
            Case CVErr(xlErrNA)
            Resultat = "#N/A"
            Case CVErr(xlErrName)
            Resultat = "#NOM?"
            Case CVErr(xlErrNull)
            Resultat = "#NULL!"
            Case CVErr(xlErrNum)
            Resultat = "#NOMBRE!"
            Case CVErr(xlErrRef)
            Resultat = "#REF!"
            Case CVErr(xlErrValue)
            Resultat = "#VALEUR!"
            End Select

            MsgBox "Il y a une erreur de type " & Resultat _
                & " dans la formule de la cellule " & Valeur.Address
        End If
    Next

    Message = MsgBox("Voulez vous ouvir l'aide en ligne Excel?", _
        vbYesNo, "Informations complementaires sur les types d'erreur")

    'Adaptez le nom du fichier et l' HelpContextId en fonction de la version d'excel
    If Message = vbYes Then Application.Help "XLMAIN10.chm", 60309 'Excel2002
    'If Message = vbYes Then Application.Help "XLMAIN09.chm", 60309'pour Excel2000
    End Sub
```

Placez cette macro dans un module standard pour supprimer les flèches d'audit:

(Ou Menu Outils/Audit de formules/Supprimer toutes les flèches)

Vba
```
    Sub effacerFlechesAudit()
    Worksheets("Feuil1").ClearArrows
    End Sub
```

III-D-3 - Retrouver les cellules qui contiennent des erreurs

La macro suivante retrouve les cellules qui contiennent des erreurs dans la plage spécifiée.

Vba
```
    Sub identifierCellulesContenantErreurs()
        Dim Plage As Range, Cible As Range
```

```vba
        'Recherche dans la plage A1:A20
        Set Plage = Range("A1:A20")

        On Error Resume Next
        Set Cible = Plage.SpecialCells(xlCellTypeFormulas, xlErrors)
        If Not Cible Is Nothing Then MsgBox Cible.Address(0, 0)
    End Sub
```

III-D-4 - La fonction Error

La fonction **Error** renvoie le message correspondant au numéro d'erreur spécifié.

Si le numéro d'erreur n'est pas valide, une erreur se produit. Si le numéro d'erreur est omis, le message correspondant à l'erreur d'exécution la plus récente est renvoyé. Si aucune erreur d'exécution n'est survenue, ou si la valeur est égale à 0, la fonction **Error** renvoie une chaîne de longueur nulle ("").

La chaîne renvoyée par la fonction **Error** correspond à la propriété "Description" de l'objet **Err**.

```vba
    MsgBox Error(9)
```

III-D-5 - La fonction IsError

La fonction **IsError** renvoie une valeur de type Boolean qui indique si l'expression spécifiée est une valeur d'erreur.

```vba
    Sub MessageSiErreurFormuleDansCellule()
        If IsError(Range("A1")) = True Then _
            MsgBox "Il y a une erreur dans la cellule."
    End Sub
```

III-D-6 - La fonction CVErr

La fonction **CVErr** renvoie une donnée de type Variant et de sous-type Error contenant un numéro d'erreur spécifié par l'utilisateur.

Cet exemple supprime la ligne complète si une cellule de la colonne A contient une erreur type #DIV/0!.

Vba

```
Sub suppressionLigne_SiErreur_Div0()
    Dim x As Integer

    For x = 20 To 1 Step -1
        'Vérifie si les cellules de la colonne A contiennent une erreur
        If WorksheetFunction.IsError(Range("A" & x)) = True Then
            'Supprime la ligne s'il s'agit d'une erreur #DIV/0!
            If CVErr(xlErrDiv0) = Range("A" & x) Then _
                Rows(x).EntireRow.Delete
        End If
    Next x
End Sub
```

IV - Conclusion

Il est particulièrement interessant de prendre en compte les erreurs lorsque vous devez gérer et anticiper toutes les manipulations de l'utilisateur final. Dans ce cas, une réflexion doit être menée en amont pour identifier les problèmes qui pourraient survenir. Faites tester le classeur par plusieurs utilisateurs (de préférence non spécialistes) et possédant des configurations différentes afin de valider votre gestionnaire d'erreur.

V - Liens

Ecrire un gestionnaire d' erreurs par **GilMir**.

Translating Automation Errors for VB VBA.

VI - Téléchargement