Tutoriel 16 : Le gestionnaire d'erreurs

par Gilbert Miralles (gilmir.developpez.com)

Date de publication : Lundi 18 mars 2003

Dernière mise à jour : Lundi 4 février 2008

I - Ecrire un gestionnaire d'erreurs

I-1 - Principe général

Erreur d'exécution

Erreur survenant lors de l'exécution du code. Une erreur d'exécution se produit lorsqu'une instruction tente d'accomplir une opération non valide.

Nous avons tous utilisé dans notre activité de développement l'instruction :

On Error Resume Next

C'est en effet une instruction intéressante car si une erreur se produit, le programme ignore l'erreur et passe à la ligne suivante.

Mais, à mon avis c 'est une instruction que l'on doit insérer dans son programmes que lorsque l'on a débugué son application et pour prévenir l'erreur imparable que l'on n'avait pas prévue lors des divers contrôles des lignes de code.


Le gestionnaire d'erreurs sert à fournir à l'utilisateur les possibilités de récupérer proprement les erreurs d'exécution du programme.

Vous utilisez généralement l'instruction **Resume** lorsque le gestionnaire d'erreurs peut corriger une erreur et l'instruction **Resume Next** dans le cas contraire.

Vous pouvez cependant écrire un gestionnaire d'erreurs qui ne révèle pas à l'utilisateur l'existence d'une erreur d'exécution ou qui affiche des messages d'erreur tout en permettant à l'utilisateur de saisir des corrections.

I-2 - Exemple pratique

La base de votre gestionnaire d'erreurs est l'instruction **On Error** :

- On Error GoTo SortieDiscrete (L'étiquette qui suit le mot clé On Error GoTo est laissée libre à votre discrétion)
- On Error Resume Next
- On Error GoTo 0

Quand Visual Basic traite l'instruction On Error, il initialise un traitement d'événements qui vérifie continuellement les conditions d'erreurs pendant que le code s'exécute.

Quand une erreur se produit, le chemin précis du déroulement du programme dépend de l'utilisation de l' instruction On Error.

La liste suivante montre les différentes expressions de cette instruction :

On Error GoTo suivi de l'étiquette que vous avez déclarée, en l'occurence dans cet exemple, nous emploierons :

On Error GoTo SortieDiscrete : le contrôle passe à la première ligne exécutable du code qui suit l'étiquette indiquée par SortieDiscrete.

Exemple :

```
Private Sub Form_Load()
 On Error GoTo Sortiediscrete
 'lignes de code du programme
 '
 '
 'Tout s' est bien passé, nous sortons de la procédure
Exit Sub

'Voici l' étiquette du gestionnaire d' erreurs
SortieDiscrete:
 'On peut mettre un message du genre(Voir en fonction de l' erreur détectée)
 msg = "Une erreur involontaire s' est produite"
 msg + msg = " veuillez relancer le programme!"
 MsgBox msg

End Sub
```

Explications :

Vous avez indiqué au programme que, s'il rencontre une erreur, qu'il veuille bien se diriger vers l'étiquette réservée à cet effet.

1er cas :

Il y a effectivement dans le listing du code une erreur inconnue provoquée par le programme ou par le système.

Dés que l'erreur est détectée, le contrôle est dirigé automatiquement vers l'étiquette Label, - dans notre cas, c'est l'étiquette "Sortie discrète" - Il l'ignore mais lit les lignes de codes qui suivent immédiatement celle-ci.

Si vous n'avez pas de message, le programme suit son cours, si vous avez un message, il le lit, ferme la page en cours, termine son cycle et attend les instructions suivantes.

2e cas : Il n'y a pas d'erreurs, le contrôle lit les lignes de code, arrive à l'instruction **Exit Sub**, ferme normalement la page en cours, sort de la procédure et attend de nouvelles instructions.

Si nous n'avions pas mis l'instruction **Exit Sub**, le contrôle aurait continué à lire les instructions suivantes et aurait affiché les les lignes de code qui se trouvent après l'étiquette Label, ce qui vous aurait induit en erreur.

Vous avez constaté que j'ai mis juste après l'étiquette le double point ":", c'est tout simplement la caractère qui indique à VB que j'ai utilisé une étiquette de renvoi.

On Error Resume Next :

Le contrôle passe à la ligne qui suit celle qui a provoquée l'erreur

On Errot GoTo 0 :

L'erreur est neutralisée et le gestionnaire d'erreurs est contourné dans la procédure.

Le code suivant montre l'utilisation d'un gestionnaire simple pour contrôler la suppression d' un fichier LANGLADE.TXT sur le lecteur de disquette.

```
Sub Main( )
 On Error GoTo InformeMoi
 Kill "A:\LANGLADE.TXT"
 'Si tout s'est passé normalement nous sortons de la procédure
 Exit Sub
 InformeMoi:
 MsgBox "Erreur de lecture de disquette"
End sub
```

Faites l'essai la première fois sans disquette et la deuxième fois avec une disquette dans laquelle vous aurez renommé un fichier existant en "Langlade.txt"

Vérifiez les attributs du fichier de façon qu'il puisse être effaçable.

Attention, ce fichier va être effacé par votre réalisation

II - Les propriétés de l'objet ERR

Une fois que l'exécution du programme a été transmise à votre routine de gestion d'erreurs, votre code peut déterminer l'erreur qui s'est produite en affichant son numéro et sa description.

Quand une erreur se produit, les informations la concernant sont stockées dans l'objet ERR.

Vous pouvez utiliser deux propriétés de cet objet, la propriété Number et la propriété Description.

Dans les vieilles versions du Basic, les instructions Error et Error() étaient utilisées.

- Err.Number : indique le numéro de l'erreur
- Err.DEscription : indique la description de l'erreur

Le code qui suit montre un exemple de code qui vérifie l'existance d'un fichier dans un répertoire particulier.

En cas d'erreur, le programme affiche le numéro de l'erreur ainsi que sa description.

```
Sub Main ( )
On Error GoTo ErreurFatale
If Dir  ("C:\Gilou\Langlade.txt") = False Then
     MsgBox "Ce fichier n' existe pas"
End If
Exit Sub
ErreurFatale:
 MsgBox "N° erreur:"&Err.Number & vbLf & Err.DEscription
 End Sub
```

La figure qui suit montre le résultat obtenu issu de la capture de l'erreur par le gestionnaire.

Le gestionnaire d'erreur permet d'afficher le numéro de l'erreur et sa description.

Vous pouvez utiliser toutes les valeurs disponibles en consultant l'aide en ligne de visual Basic en appuyant sur la touche F1 de votre clavier.

Prochain tutoriel :

Technique pour réaliser un Password pour Windows