

Application multilingue

par Freak Comtois

Date de publication : 02 juillet 2008

Dernière mise à jour :

Cet article présente la méthode utilisée pour gérer les langues dans l'EDI de PureBasic. Il s'agit pour l'essentiel d'une traduction d'un article écrit par Freak l'auteur de l'EDI, que j'ai complétée en détaillant un peu plus certains passages.

I - Introduction.....	3
II - Exemples fichiers.....	3
III - DataSection.....	4
IV - Procédure LoadLanguage.....	4
IV-1 - Comptage.....	5
IV-2 - Chargement de la langue par défaut.....	5
IV-3 - Tri et indexation groupe.....	6
IV-4 - Chargement langue externe.....	6
V - Procédure Language.....	7
V-1 - Contrôle Groupe.....	7
V-2 - Recherche mot.....	7
VI - Exemple d'utilisation.....	8
VII - Lien.....	8
VIII - Remerciements.....	8

I - Introduction

Cet article présente la méthode utilisée dans l'EDI de PureBasic pour gérer les langues. L'EDI utilise un peu plus de code pour mémoriser d'autres informations contenues dans les fichiers (traducteurs, etc), mais le code de base est identique.

L'ancienne méthode utilisée était un simple tableau de chaînes de caractères appelées par leur index. Il s'agit bien sûr d'une bonne solution pour un petit programme, mais cela est devenu un problème à mesure que le projet grandissait. Le tableau des index n'est pas descriptif de ce qu'il contient, et il est donc difficile de dire quelle chaîne de caractères sera insérée dans le code. En fait, c'est aussi un cauchemar pour ajouter de nouvelles chaînes, en particulier avec la mise à jour de fichiers externes de traduction. Le code que vous voyez ici (celui qui est en service depuis 3,94) a prouvé que la gestion des langues est très simple et il permet de résoudre les problèmes mentionnés ci-dessus. Le code se compose de deux procédures :

- LoadLanguage() pour charger une langue.
- Language() pour utiliser cette langue dans le programme.

Bénéfice de cette solution :

- Les chaînes de caractères sont identifiées par un groupe et un nom clé, ce qui permet de mieux les organiser avec leurs propres descriptions et de développer plus facilement.
- Les chaînes de caractères sont indexées et triées lors de leur chargement, ce qui permet un accès rapide malgré qu'elles soient appelées par leur nom.
- Une langue par défaut est définie dans le code (**DataSection**), ce qui permet d'avoir toujours un texte par défaut dans le cas où un fichier externe est manquant ou périmé.
- La liste de mots est facile à élargir. Il suffit d'ajouter une nouvelle entrée dans la DataSection ainsi que le fichier de langue et d'utiliser le nouveau groupe et nom clé.
- Les fichiers de langues additionnels sont dans le format "PB **Preference**" qui permet de les maintenir facilement.

Utilisation :

- Définir le langage par défaut dans la DataSection comme montré plus bas
- Utiliser LoadLanguage() au départ pour charger le langage par défaut ou le fichier externe.
- Utiliser Language(Group\$, Name\$) pour accéder aux mots des langues.

II - Exemples fichiers

Exemples de fichiers langue issus directement de l'EDI de PureBasic.

Le nom d'un groupe se trouve entre crochets, par exemple [Compiler]. Le nom clé est identique quelque soit le fichier langue, c'est la description de ce mot qui sera traduite. et enfin c'est ce mot clé que vous utiliserez dans votre programme. Considérez le comme un mnémonique sur le texte que vous souhaitez afficher.


Anglais	Français
[MenuTitle]	[MenuTitle]
File = &File	File = &Fichier
Edit = &Edit	Edit = &Edition
Compiler = &Compiler	Compiler = &Compilateur
Debugger = &Debugger	Debugger = &Débogueur

Tools = &Tools Help = &Help	Tools = &Outils Help = &Aide
[Compiler] OptionsTitle = Compiler Options MainFile = Main source file UseIcon = Use Icon EnableDebugger = Enable Debugger EnableASM = Enable inline ASM support EnableXP = Enable XP skin support	[Compiler] OptionsTitle = Options du compilateur MainFile = Fichier principal UseIcon = Utiliser une icône EnableDebugger = Activer le débogueur EnableASM = Activer l'assembleur en ligne EnableXP = Activer le support des thèmes XP

III - DataSection

C'est ici que la langue par défaut est définie. C'est une liste de Groupe, suivi d'une liste de noms, par exemple dans le groupe [MenuItem], on trouve le nom 'Open', avec une description plus complète à droite, c'est cette description qu'il est possible de traduire dans un fichier et qui sera affichée dans votre programme
Avec quelques mot-clés spéciaux pour le groupe :

- "_GROUP_" indique un nouveau groupe dans la DataSection, la seconde valeur est le nom du groupe.
- "_END_" indique la fin de la liste de la langue (Il n'y a pas de nombre de mots prédéfinis, le comptage est effectué dans la procédure **LoadLanguage()**).

 Les index de mot sont insensibles à la casse pour se faciliter la vie.

```

DataSection
Language :

; =====
Data$ "_GROUP_",      "MenuTitle"
; =====

Data$ "File",         "File"
Data$ "Edit",         "Edit"

; =====
Data$ "_GROUP_",      "MenuItem"
; =====

Data$ "New",          "New"
Data$ "Open",         "Open..."
Data$ "Save",         "Save"

; =====
Data$ "_END_",        ""
; =====

EndDataSection

```

IV - Procédure LoadLanguage

Cette procédure charge le langage par défaut ou une langue à partir d'un fichier. Elle doit être appelée au moins une fois avant d'utiliser n'importe quel mot.

Cette procédure permet de :

- Charger et trier le langage par défaut inclus dans le programme.
- Charger n'importe quel langage à partir d'un fichier.

Cette méthode vous permet d'obtenir un mot d'une langue, même si le fichier n'est pas trouvé, ou qu'un mot référencé par un index est introuvable dans le fichier. Vous récupérez la langue par défaut en utilisant la procédure Language().

Cette procédure peut être appelée plusieurs fois pour changer la langue utilisée durant l'exécution de votre programme.

IV-1 - Comptage

Dans un premier temps, la procédure effectue un comptage du nombre de groupes, et du nombre de chaînes (Chaque ligne est composée d'un nom clé (mnémonique) et d'une description) contenus dans la langue par défaut incluse dans la **DataSection**.

```
NbLanguageGroups = 0
NbLanguageStrings = 0

Restore Language
Repeat

    Read Name$
    Read String$

    Name$ = UCase(Name$)

    If Name$ = "_GROUP_"
        NbLanguageGroups + 1
    ElseIf Name$ = "_END_"
        Break
    Else
        NbLanguageStrings + 1
    EndIf
Forever
```

Une fois le comptage terminé, les tableaux sont déclarés.

```
Global Dim LanguageGroups.LanguageGroup(NbLanguageGroups) ; ils sont tous là
Global Dim LanguageStrings.s(NbLanguageStrings)
Global Dim LanguageNames.s(NbLanguageStrings)
```

IV-2 - Chargement de la langue par défaut

La procédure charge la langue standard (par défaut) se trouvant dans la **DataSection**.

```
Group = 0
StringIndex = 0

Restore Language
Repeat

    Read Name$
    Read String$

    Name$ = UCase(Name$) ; insensible à la casse

    If Name$ = "_GROUP_"
        LanguageGroups(Group)\GroupEnd = StringIndex
        Group + 1

        LanguageGroups(Group)\Name$ = UCase(String$) ; insensible à la casse
        LanguageGroups(Group)\GroupStart = StringIndex + 1
    For i = 0 To 255
```

```

        LanguageGroups(Group)\IndexTable[i] = 0
    Next i

    ElseIf Name$ = "_END_"
        Break ; Il n'y a plus de données à lire

    Else
        StringIndex + 1
        LanguageNames(StringIndex) = Name$ + Chr(1) + String$
        ; Garde le nom de l'index et la valeur ensemble pour trier plus facilement

    EndIf

    ForEver

    LanguageGroups(Group)\GroupEnd = StringIndex ; Configure la fin pour le dernier groupe !
    
```

IV-3 - Tri et indexation groupe

La procédure effectue l'indexation et un tri pour chaque groupe.

La bibliothèque **String** est grandement utilisée ici.

SortArray() permet de trier le tableau pour chaque groupe, selon les valeurs début et fin d'index du groupe en cours de traitement.

Pour accélérer les recherches dans la procédure **Language()**, chaque groupe possède une table permettant de retrouver un index rapidement en fonction de la première lettre du nom clé, **LanguageGroups(Group)\IndexTable[char] = StringIndex**.

```

    For Group = 1 To NbLanguageGroups
        If LanguageGroups(Group)\GroupStart <= LanguageGroups(Group)\GroupEnd

            SortArray(LanguageNames(), 0, LanguageGroups(Group)\GroupStart, LanguageGroups(Group)\GroupEnd)

            char = 0
            For StringIndex = LanguageGroups(Group)\GroupStart To LanguageGroups(Group)\GroupEnd
                LanguageStrings(StringIndex) = StringField(LanguageNames(StringIndex), 2,
                Chr(1)) ; Sépare la valeur de l'index
                LanguageNames(StringIndex) = StringField(LanguageNames(StringIndex), 1, Chr(1))

                If Asc(Left(LanguageNames(StringIndex), 1)) <> char
                    char = Asc(Left(LanguageNames(StringIndex), 1))
                    LanguageGroups(Group)\IndexTable[char] = StringIndex
                EndIf
            Next StringIndex

        EndIf
    Next Group
    
```

IV-4 - Chargement langue externe

La procédure tente de charger un fichier de langue externe.

La bibliothèque **Preference** rend de grands services, notamment ici avec la fonction **ReadPreferenceString()** qui permet de gérer simplement les valeurs par défaut. En effet, **LanguageStrings(StringIndex)** contiendra toujours une valeur valide, qui sera issue soit du fichier demandé, soit de la langue par défaut en **DataSection**.

```

    If FileName$ <> ""

        If OpenPreferences(FileName$)
            For Group = 1 To NbLanguageGroups
                If LanguageGroups(Group)\GroupStart <= LanguageGroups(Group)\GroupEnd
                    PreferenceGroup(LanguageGroups(Group)\Name$)

                    For StringIndex = LanguageGroups(Group)\GroupStart To LanguageGroups(Group)\GroupEnd
                    
```

```

        LanguageStrings(StringIndex) = ReadPreferenceString(LanguageNames(StringIndex),
LanguageStrings(StringIndex))
        Next StringIndex
    EndIf
Next Group
ClosePreferences()

    ProcedureReturn #True
EndIf

EndIf
    
```

V - Procédure Language

Cette procédure retourne un mot pour la langue en cours d'utilisation. Chaque mot est identifié par un groupe et un nom clé(les deux sont insensibles à la casse).

Si le nom n'est pas trouvé (ou non inclus dans la langue par défaut) le retour sera "##### String not found! #####". Cela aide à trouver les erreurs dans le code de la langue facilement.

V-1 - Contrôle Groupe

Dans un premier temps la procédure vérifie que le groupe demandé existe. S'il existe son numéro est mémorisé dans la variable 'Group' qui est statique afin de conserver sa valeur au prochain appel de la procédure.

```

Static Group.1 ; Pour un accès plus rapide quand on utilise le même nom de groupe plusieurs fois
Protected String$, StringIndex, Result

Group$ = UCase(Group$)
Name$ = UCase(Name$)
String$ = "##### String not found! #####" ; Pour aider à trouver les erreurs

If LanguageGroups(Group)\Name$ <> Group$ ; Contrôle si c'est le même groupe à chaque fois
    For Group = 1 To NbLanguageGroups
        If Group$ = LanguageGroups(Group)\Name$
            Break
        EndIf
    Next Group

    If Group > NbLanguageGroups ; Le groupe demandé n'est pas trouvé
        Group = 0 ; Initialise le numéro de groupe
    EndIf
EndIf
    
```

V-2 - Recherche mot

Si le groupe demandé existe, la recherche de la chaîne correspondant au nom clé peut commencer.

Pour réduire la recherche, les noms clés sont **triés**, ce qui fait que la boucle de recherche débute au premier index correspondant à la première lettre du nom recherché.

```

If Group <> 0
    StringIndex = LanguageGroups(Group)\IndexTable[ Asc(Left(Name$, 1)) ]
    If StringIndex <> 0

        Repeat
            Result = CompareMemoryString(@Name$, @LanguageNames(StringIndex))

            If Result = #PB_String_Equal
                String$ = LanguageStrings(StringIndex)
                Break

            ElseIf Result = #PB_String_Lower ; Mot non trouvé
                
```

```
Break

EndIf

StringIndex + 1
Until StringIndex > LanguageGroups(Group)\GroupEnd

EndIf

EndIf

ProcedureReturn String$
```

VI - Exemple d'utilisation

Vous trouverez le détail du fichier **german.prefs** dans la page [Source source](#).

```
LoadLanguage() ; charge la langue par défaut
;LoadLanguage("german.prefs") ; décommentez pour charger le fichier de langue allemand

; Récupère quelques mots de la langue
;
Debug Language("MenuTitle", "Edit")
Debug Language("MenuItem", "Save")
```

VII - Lien

Consultez le [Source code source complet](#).

VIII - Remerciements

Je tiens à remercier tout particulièrement **Timo 'fr34k' Harter** pour m'avoir autorisé à utiliser tout ses articles et notamment celui ci, **CORBASE** pour sa participation à la traduction de cet article, et **gorgonite** pour la relecture de l'article, sans oublier l'équipe PureBasic qui travaille sans relâche pour améliorer ce fabuleux langage.