MCL_INI : TROUSSE DU PROGRAMMEUR

Compatibilité MultiLingue dans les outils MapInfo avec la solution MLC_INI[1]

Documentation et ressources pour le programmeur en MapBasic

Jacques Paris, Juillet 2000

jakesp@total.net

[1] Le projet de Compatibilité MultiLingue est une initiative de Jacques PARIS (jakesp@total.net). La SOLUTION MLC_INI est l'inspiration de Bill THOEN (bthoen@ctmap.com); elle a été mise au point avec l'aide de Mats Elfström (mats.elfstrom@telia.com) et de Carlos Montalvillo Gómez (Carlos.Montalvillo@sgsmap.com)

Le fichier MlcInKit.zip contient une variété de documents et des exemples. Ils comprennent en particulier les bibliothèques de ressource requises pour compiler et assembler un projet respectant la solution standard MLC_INI, des exemples documentés et ce document.

Bibliothèques MLC_INI

mlc_ini.def	définie les routines et fonctions de la bibliothèque mlc_ini.mb l
mlc_ini.mb	contient les routines et fonctions correspondant aux appels faits dans l'application et traite de toutes les fonctions relatives àla langue
profile.def	définie les fonctions API pour lire/écrire le fichier ini
profile.mb	nécessaire sous sa forme MBO pour assembler un projet

Ces fichiers devraient être places dans un répertoire central de façon àn'en conserver qu'une seule copie; les références àces fichiers devraient comprendre l'adresse complète de ce répertoire.

PROJECT FILE

Chaque application requière un fichier projet (*application*.mbp) de façon àassembler les bibliothèques ressources au module principal de l'application (et les différents modules de l'application ensemble, s'il y en a plus d'un). Son contenu type est :

[LINK]

application=*application*.mbx	nom de l'application mbx
module=*application*.mbo	module principal de l'application (plusieurs modules d'application sont parfois nécessaires)
module= MLC_INI.mbo	choix de langue, sub/fonction de gestion de textes
module=profile.mbo	fonctions API de Windows

ÉLÉMENTS DE PROGRAMMATION

1 – Énoncé(s) Include

> ***note**: spécifier l'adresse complète de ces fichiers s'ils ne résident pas dans le même répertoire où l'application est développée.*

Toujours requis

```
include "mlc_ini.def"
```

Requis seulement si on fait des appels directs aux sub/function API (ini_GetIni, ini_WriteINI)

```
include "profile.def"
```

2 – Énoncés Declare

Emsemble minimal

(Goodbye et About sous une forme ou une autre sont recommandés, ils ne sont pas requis)

```
declare sub main
declare sub MenuSetup
declare sub option_lang
declare sub goodbye
declare sub about
declare Sub BuildLanguage (byval sProfile
                           as string)
```

3 – Initialisation du langage

Cette routine détecte si le fichier .INI existe. S'il n'y en a pas, elle en crée un avec `BuildLanguage()`.

Elle enregistre la langue àutiliser (la langue originale si un nouveau .INI est créé, ou la langue choisie précédemment si un .INI existe) et lance la préparation des menus.

Si d'autres paramètres sont maintenus dans le fichier .INI, c'est l'endroit où lire leurs valeurs avec la fonction ini_GetINI().

```
      dim fileini as string
'=============================================
      sub main

      fileini=Applicationdirectory$()+
                           "APPLICATION.ini"
      if not FileExists (fileini) then
           call BuildLanguage (fileini)
      end if
      call mlc_InitLanguage (fileini)
      call MenuSetup

      end sub
```

4 –Création de menus

Les numéros des messages n'ont d'autre sens ici que de remplir les espaces définis. Installer l'application dans le menu OUTILS (ID 4) évite d'encombrer la barre du menu principal par toutes les applications ouvertes. Cette possibilité n'est offerte que pour les versions =>4.5

Cette routine devrait aussi contenir si nécessaire les définitions des menus « en contexte » (shortcut menus), les modifications aux barres d'outils (ajout d'icônes, par exemple) et la création de nouvelles barres d'outils.

```
      sub MenuSetup

      Create Menu msg(8) as
           msg(14) calling option_lang,
           msg(15) calling options,
           "(-",
           msg(9)  Calling About,
           msg(10) Calling Goodbye

      if systeminfo(3)<450 then
           Alter Menu Bar Add msg(8)
        else
           Alter Menu ID 4 Add msg(8) As msg(8)
      end if
      end sub
```

5 – Changement de language

La fonction **`mlc_SetLanguageDlg()`** permet de choisir parmi les langages présents dans le fichier .INI.

Si d'autres menus ou barres d'outils sont impliqués par le changement de langue, ils devraient être traités dans cette routine; les items de menu peuvent être simplement retirés (remove) mais les barres d'outils doivent être détruits (destroy).

```
      sub option_lang

      dim sPrevMenu,a as string

      sPrevMenu = msg(8)
      if mlc_SetLanguageDlg () then
           if systeminfo(3)<450 then
                Alter menu bar remove sPrevMenu
             else
                a="Alter menu ID 4 remove
                           """+sprevmenu+"""
                run command a
           end if
           call MenuSetup
      end if
      end sub
```

6 – Écriture des "phrases"

Les phrases correspondant aux “marqueurs de texte” (paramètres d’appel àla fonction msg(nn)) sont inscrites dans une routine spéciale dont il faut respecter la structure et les règles d’écriture.

La taille du vecteur doit être ajustée pour correspondre au nombre exact de phrases.

L’appel à**`mlc_PrimeLanguage()`**enregistre les phrases en langue originale dans le fichier .INI. « English » doit être remplacé par le nom de la langue si elle est différente.

```
Sub BuildLanguage(byval sProfile as string)
dim sMsg(23) as string

sMsg(1) = "\nDirect Access to Preferences"
sMsg(2) = "\nSystem Settings"
sMsg(3) = "\nMap Window"
...
sMsg(22) = "Effective only at the next
                                  loading"
sMsg(23) = "of the application"

call mlc_PrimeLanguage ("English", sMsg,
                                  sProfile)
End Sub
```

7 – Appels directs aux ressources Profile: utilisation du fichier .INI pour conserver des paramètres d’application

Si le programmeur veut conserver dans le fichier .INI des paramètres propres àson application, il devra utiliser les ressources de la bibliothèque Profile pour écrire/lire ce fichier. Profile.def doit alors être inclus dans le code de l’application (voir §1-). Ces énoncés ont la syntaxe suivante :

sous-routine	call **ini_WriteIni** (Group, Keyname, Value, Profile)
et	
fonction	Value = **ini_GetIni** (Group, Keyname, Default, Profile)

Profile	le « nom » du fichier .INI (chaîne)
Valeur	une chaîne contenue entre les “ après le signe =.
Group	le texte encadré par les crochets [ ] (Language1)
Keyname	l’identificateur avant le signe = (msg7).
	for ini_GetIni() only
Default	valeur retournée si le Keyname dans le Group n’est pas trouvé.

Le code MB des bibliothèques utilise “?” systématiquement

Noter que les fonctions Profile marche aussi bien sur des installations Windows 16-bits ou 32-bits.

L’application PREFER démontre une telle application.

8 – Langue originale autre que l’anglais

Si le programmeur utilise une autre langue que l’anglais dans ses phrases, il doit alors traduire dans cette langue les 3 messages [sMsg(1) … sMsg(3)] de la routine `mlc_BuildLanguage()` située dans la bibliothèque mcl_ini.mb.

```
sub mlc_BuildLanguage (byval sProfile as string)

dim sMsg(3) as string
dim i as smallint
sMsg(1) = "Language option"
sMsg(2) = "Your choice will become active when
you close this dialog"
sMsg(3) = "OK"
for i = 1 to UBound(sMsg)
      call ini_WriteIni ("MLC1", "msg"+Str$(i),
                         sMsg(i), sProfile)
next
```

```
end sub
```

9 – Recommandations pour la conception des dialogues

Comme la traduction dans une autre langue peut résulter dans des phrases de longueur différente, cela devient difficile de concevoir des dialogues qui ne soit pas affectés par cette contrainte. le programmeur devra préférer à un dialogue visuellement plaisant, à un dialogue compact contenu dans un cadre unique, un dialogue « sécuritaire » ou décomposé en plusieurs cadres.

Exploser un dialogue en plusieurs cadres est probablement facile à comprendre et à mettre en œuvre, mais la notion de « sécurité » est plus difficile à imaginer. Cela veut dire que les éléments essentiels d'un dialogue ne devraient pas être affectés par des textes de longueur variable, ou que les textes mêmes ne soient pas « contraints » par des éléments fixes. Voici quelques idées pratiques à ce sujet :

- sur une même ligne, il est préférable de placer une boîte EditText avant un texte StaticText plutôt qu'après

- ne pas spécifier une largeur pour un dialogue en particulier permettra l'ajustement automatique aux longueurs de chaîne, sans les tronquer

- utiliser l'alignement à gauche seulement; oublier tout effort de centrage ou d'alignement à droite

- la position d'un élément ne devrait être spécifiée que si c'est indispensable

- hauteur et longueur variable sont en général indépendantes sauf dans le cas d'une boîte StaticText multiligne; la phrase traduite peut représenter pour la largeur donnée plus de lignes que l'originale; il y a donc risque de troncation. Il serait préférable d'utiliser plusieurs StaticText d'une seule ligne pour éviter ce problème.

AVERTISSEMENT:

Chaque fois qu'une application est lancée, un nouveau fichier INI est créé seulement s'il n'en existe pas déjà Si les « phrases » sont changées dans l'application, elle n'apparaîtront dans le fichier INI et dans l'application que si le fichier INI est recréé avec les nouvelles phrases. Il faut donc éliminer le fichier INI d'une soumission précédente avant de relancer l'application.

MODULE PRINCIPAL D'APPLICATION : un exemple PREFER.MB

Le module principal d'application doit contenir certains appels spécifiques et traiter certaines situations d'une façon spécifique. L'exemple suivant documente les éléments clés

L'information générique est détaillée dans « Éléments de programmation » (voir plus haut). Les commentaires inclus traitent surtout de la mise en œuvre dans l'application PREFER.

requis

```
include "mlc_ini.def"
```

Requis dans cette application à cause des appels directs à la bibliothèque Profile pour écrire/lire la valeur de paramètres dans le fichier .INI

```
include "profile.def"
```

Structure générale des routines

```
declare sub main
declare sub MenuSetup
declare sub option_lang
declare sub goodbye
declare sub about
declare Sub BuildLanguage (byval sProfile as string)
```

Routine spécifique pour gérer l'option supplémentaire

```
declare sub options
```

Les parameters irow,ipos ajoutés au fichier .INI pour cette application

```
dim irow,ipos as smallint
```

Fileini est une façon de simplifier l'écriture d'appels de sub/fonctions; particulièrement utile quand il y a des paramètres supplémentaires

```
dim fileini as string

'================================================
sub main
fileini=Applicationdirectory$()+"PREFER.ini"
if not FileExists (fileini) then
     call BuildLanguage (fileini)
```

Écriture des paramètres supplémentaires (ici, valeurs initiales = 6)

```
call ini_WriteIni("General","DockedPadRow","6",
                                                fileini)
call ini_WriteIni("General","DockedPadPos","6",
                                                fileini)
end if
call mlc_InitLanguage (fileini)
call MenuSetup
end sub

'================================================
sub MenuSetup
dim sCmd as string
```

Crée le menu initial ou le recrée dans une langue différente

Dans cet exemple, la partie principale de l'application est la mise en place d'un simple menu et d'une barre d'outils. Il y aurait normalement un item du menu pour lancer les opérations.

```
Create Menu msg(8) as
    msg(14) calling option_lang,
    msg(15) calling options,
      "(-",
    msg(9)  Calling About,
    msg(10) Calling Goodbye

      if systeminfo(3)<450 then
            Alter Menu Bar Add msg(8)
      else
            Alter Menu ID 4 Add msg(8) As msg(8)
      end if
```

Lecture des valeurs des paramètres supplémentaires

```
irow=ini_GetIni("General","DockedPadrow",MSG_DEFAULT,
                                                fileini)
ipos=ini_GetIni("General","DockedPadPos",MSG_DEFAULT,
                                                fileini)
```

Création de la barre d'outils ancrée dans la position irow, ipos. C'est l'essentiel de cette application.

```
create buttonpad msg(8) as
pushbutton calling about icon 230 helpmsg msg(1)
pushbutton calling 210 icon 116 HelpMsg msg(2)
pushbutton calling 212 icon 110 HelpMsg msg(3)
pushbutton calling 215 icon 109 HelpMsg msg(4)
```

```
pushbutton calling 211 icon 117 HelpMsg msg(5)
pushbutton calling 213 icon 98 HelpMsg msg(6)
pushbutton calling 214 icon 101 HelpMsg msg(7)
fixed toolbarposition (irow,ipos)
end sub

'================================================
```
sub option_lang
```
dim sPrevMenu,a as string
```
Cette routine traite seulement du changement de langue
```
sPrevMenu = msg(8)
if mlc_SetLanguageDlg () then
      if systeminfo(3)<450 then
            Alter menu bar remove sPrevMenu
        else
            a="Alter menu ID 4 remove
                              """+sprevmenu+"""
            run command a
      end if
```
La barre d'outils est carrément détruite ici car sa définition n'est pas conservée comme celle d'un menu. Elle est recréée dans la nouvelle langue par l'appel suivant.
```
      Alter ButtonPad sPrevMenu Destroy
      call MenuSetup
end if
end sub

'================================================
```
Sub options

Routine spécifique pour entrer/modifier les valeurs des paramètres supplémentaires et enregistrer les nouvelles valeurs dans le fichier .INI.
```
dim jrow,jpos as smallint
dialog title msg(16)
control staticText title msg(17)+str$(irow)
                              position 10,10
control staticText title msg(18)+str$(ipos)
                              position 20,20
control staticText title msg(19) position 10,35
control edittext value irow into jrow width 15
                              position 20,45 height 10
control edittext value ipos into jpos width 15
                              position 20,58 height 10
control staticText title msg(20) position 45,45
control statictext title msg(21) position 45,58
control statictext title msg(22) position 10,75
control statictext title msg(23) position 10,85
control okbutton
if not commandinfo(1) then exit sub end if
if jrow<>irow then
      call ini_WriteIni("General","DockedPadRow",jrow,
                                          fileini)
      irow=jrow
end if
if jpos<>ipos then
      call ini_WriteIni("General","DockedPadPos",jpos,
                                          fileini)
      ipos=jpos
end if
end sub

'================================================
Sub About
note
msg(11)+chr$(13)+chr$(13)+msg(12)+chr$(13)+chr$(13)+ms
g(13)
end sub

'================================================
Sub GoodBye
  End Program
End Sub

'================================================
Sub BuildLanguage(byval sProfile as string)
dim sMsg(23) as string
```
Toutes les phrases requises par l'application. Noter que le numérotage ne respecte pas l'ordre d'apparition dans la liste des codes; c'est l'effet des refontes multiples de l'application non suivies d'une reprise systématique des numéros d'appel.
```
      sMsg(1) = "\nDirect Access to Preferences"
      sMsg(2) = "\nSystem Settings"
      sMsg(3) = "\nMap Window"
      sMsg(4) = "\nLegend Window"
      sMsg(5) = "\nStartUp"
      sMsg(6) = "\nAddress Matching"
```

```
      sMsg(7) = "\nDirectories"
      sMsg(8) = "Preference"
      sMsg(9) = "About Preferences"
      sMsg(10) = "Remove Preferences"
      sMsg(11) = "Direct access to the various
                              Preferences requesters"
      sMsg(12) = "This ultra simple program is a
                  demonstration of a multi lingual
                  application using messages from the
                  ini file"
      sMsg(13) = "Jacques Paris under Mats Elfström's
                  influence and serious
                  rework by Bill Thoen, June 2000"
      sMsg(14) = "Language Choice"
      sMsg(15) = "ToolPad Position"
      sMsg(16) = "Position of docked ToolPad"
      sMsg(17) = "ToolPad is now docked on row "
      sMsg(18) = "and in the postion "
      sMsg(19) = "Enter the values you want for"
      sMsg(20) = "row (0 topmost row)"
      sMsg(21) = "position (0 rightmost position)"
      sMsg(22) = "Effective only at the next loading"
      sMsg(23) = "of the application"

call mlc_PrimeLanguage ("English", sMsg, sProfile)
End Sub
```