Tutorial sur les MFC

1 Avant de commencer

Il peut naturellement rester des fautes d'inattention. Si tel est le cas, veuillez m'en excuser.

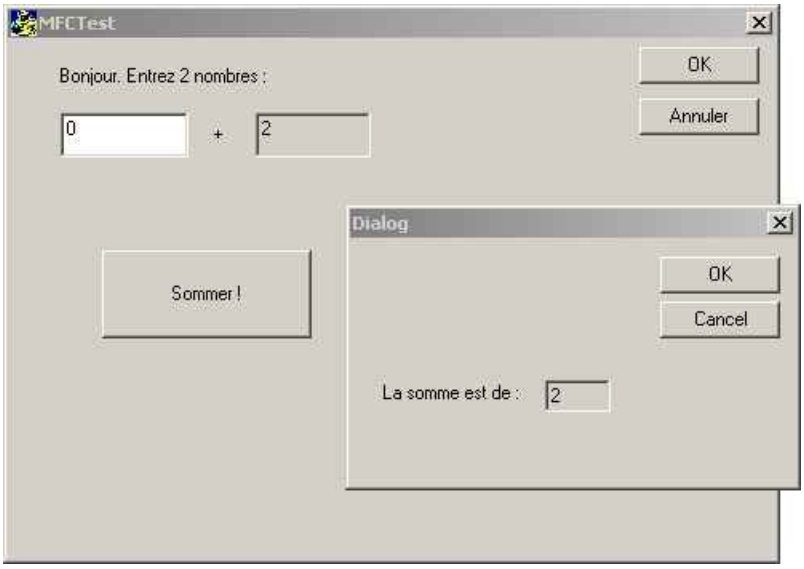
Important : ce tutorial a pour but de programmer une interface graphique de façon PRATIQUE sans tomber dans les considérations techniques les plus poussées.
Il s'adresse à un large public et je pars du principe que vous êtes assez grands pour chercher les informations complémentaires dont vous aurez besoin, tout seul.

2 Outils

Microsoft Visual C++

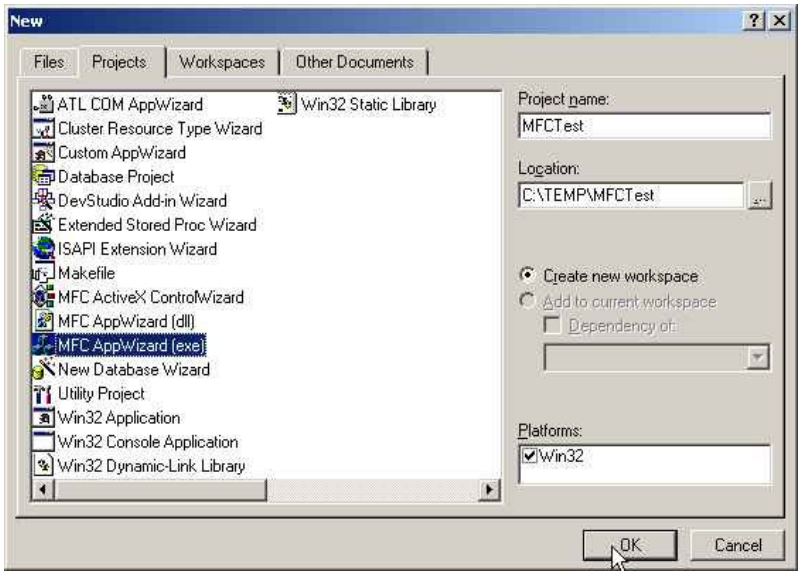
3 But

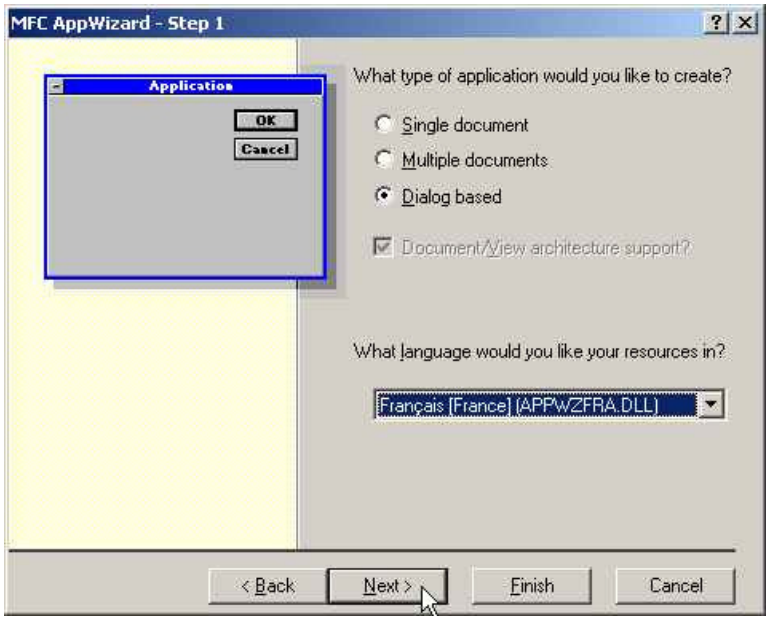
Nous voulons donc réaliser une interface graphique permettant d'additionner deux chiffres. C'est totalement inutile, sauf dans le cas de ce document :

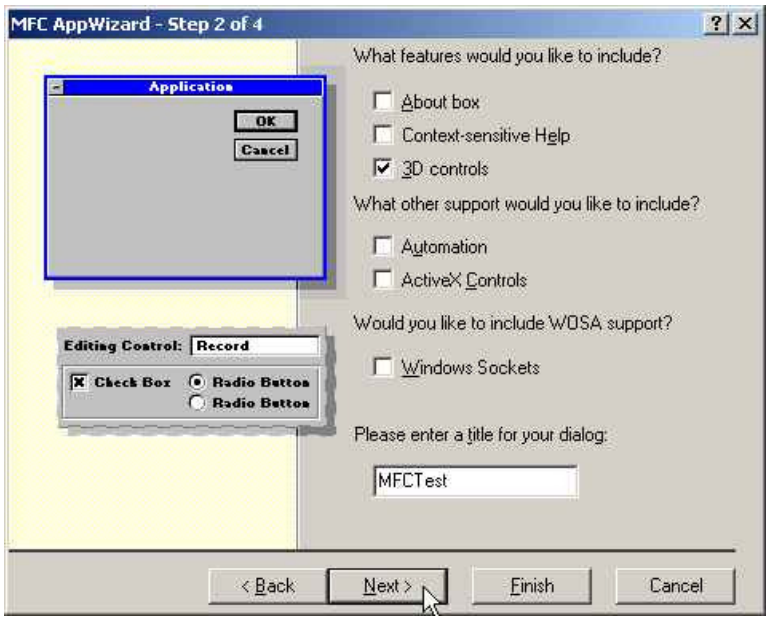


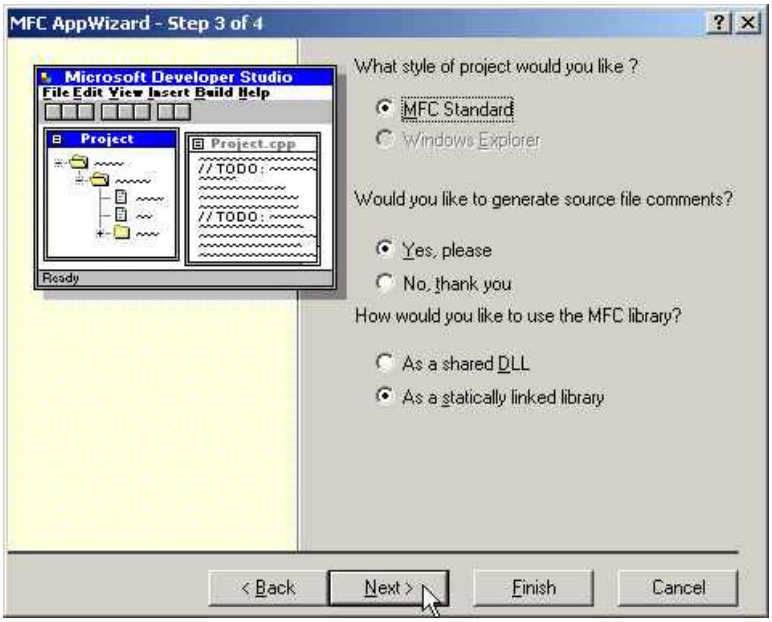
4 Réalisation

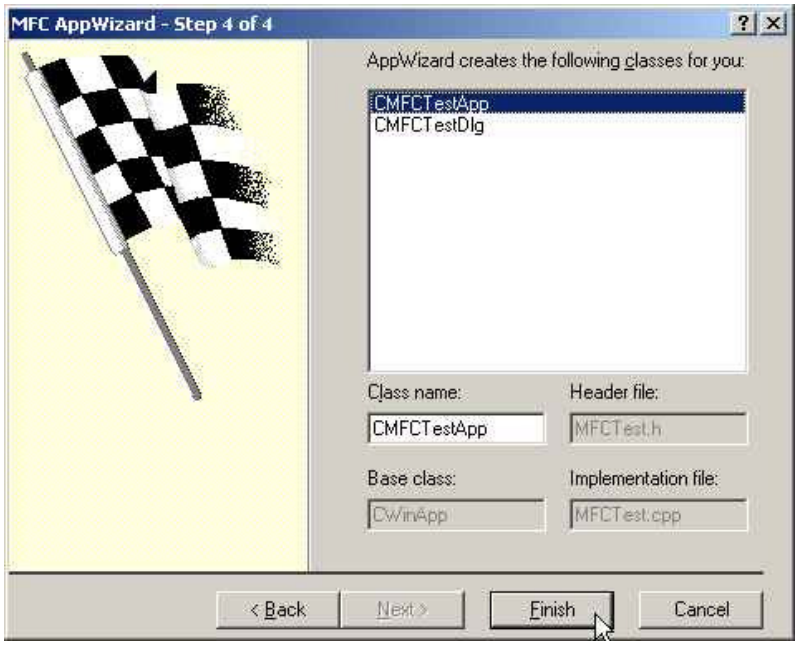
4.1 Création du projet




MFC AppWizard - Step 1
Application
OK
Cancel
What type of application would you like to create?
Single document
Multiple documents
Dialog based
Document/View architecture support?
What language would you like your resources in?
Français [France] (APPWZFRA.DLL)
< Back
Next >
Finish
Cancel


MFC AppWizard - Step 2 of 4
Application
OK
Cancel
Editing Control:
Record
Check Box
Radio Button
Radio Button
What features would you like to include?
About box
Context-sensitive Help
3D controls
What other support would you like to include?
Automation
ActiveX Controls
Would you like to include WOSA support?
Windows Sockets
Please enter a title for your dialog:
MFCTest
< Back
Next >
Finish
Cancel

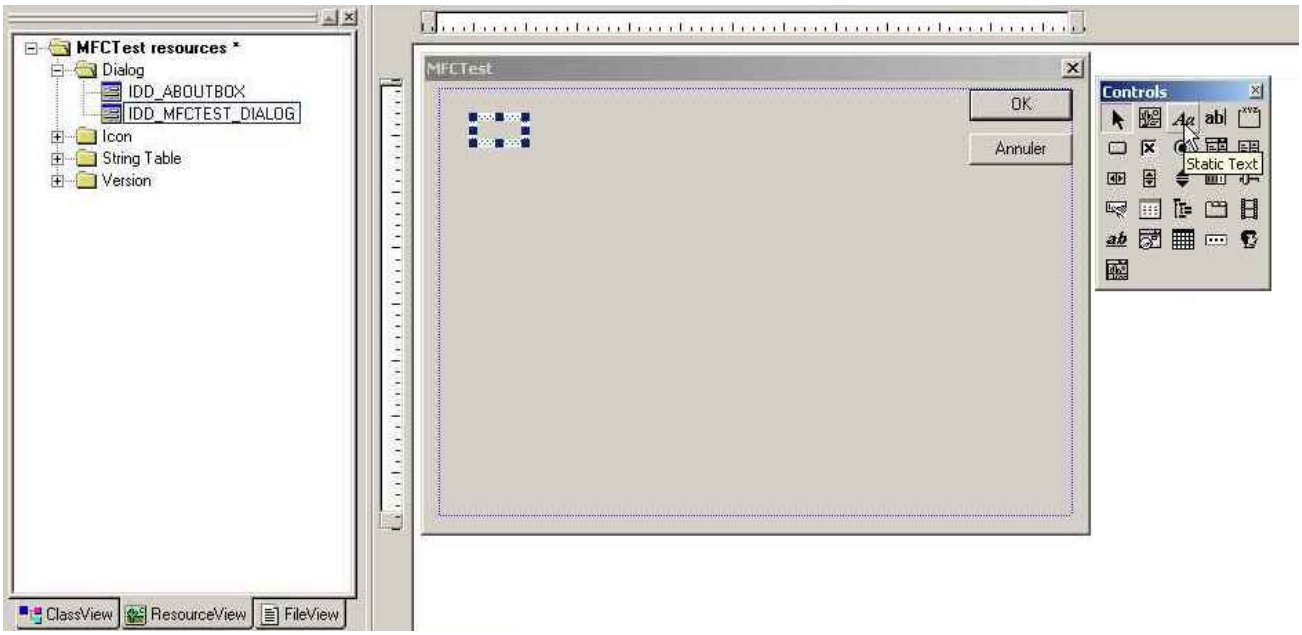

MFC AppWizard - Step 3 of 4
Microsoft Developer Studio
File Edit View Insert Build Help
Project
Project.cpp
//TODO:
//TODO:
Ready
What style of project would you like ?
MFC Standard
Windows Explorer
Would you like to generate source file comments?
Yes, please
No, thank you
How would you like to use the MFC library?
As a shared DLL
As a statically linked library
< Back
Next >
Finish
Cancel



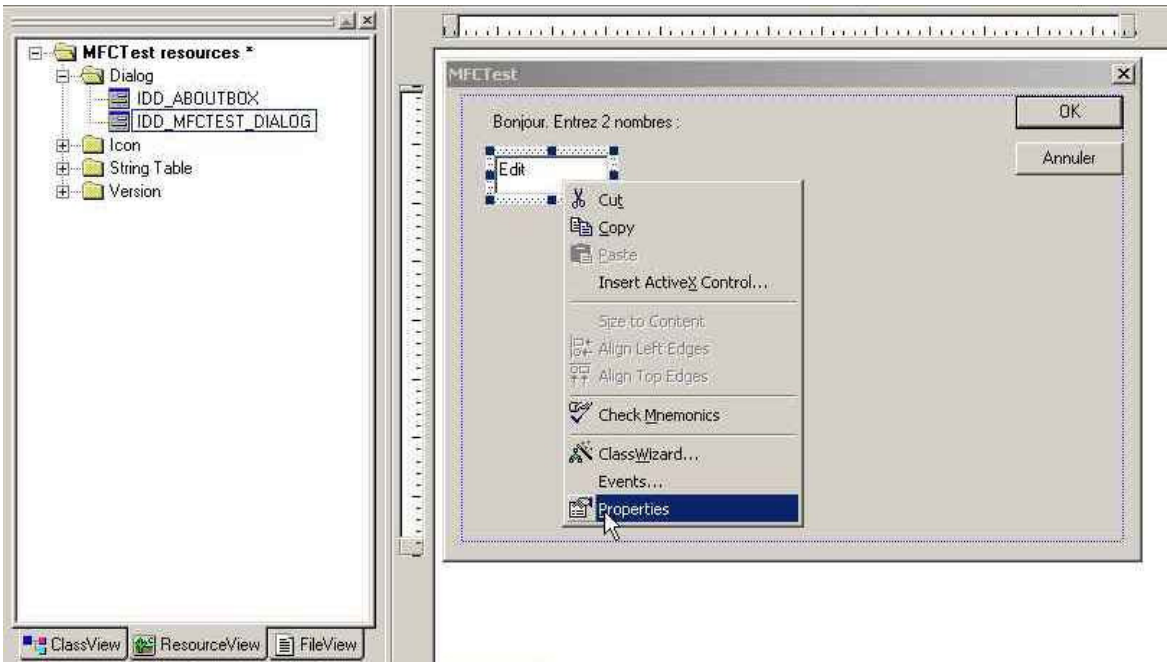
4.2 Création de l'interface graphique

Pour vous aider, faîtes apparaître la barre nommée « Control ». Un simple glisser-déplacer vous suffira alors pour mettre en place les éléments de l'interface.

Insérons un texte d'explication :

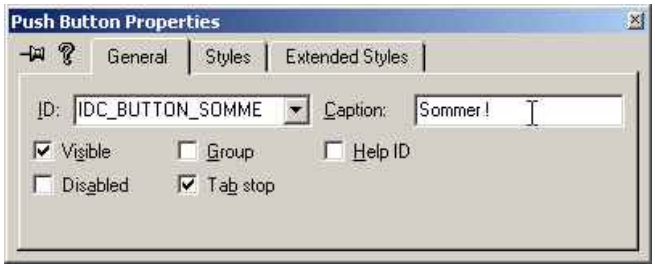


ajoutons un champ qui recevra le premier chiffre :



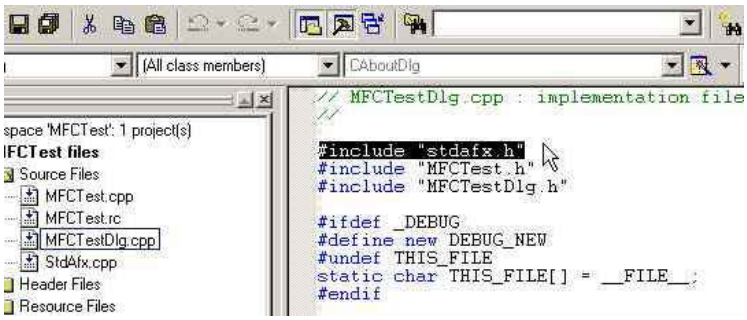
et définissons lui des propriétés (ici, donnons lui un nom clair. C'est surtout utile en présence de nombreux champs pour s'y retrouver) par un clic-droit➔Propriétés :

ajouter de la même façon le deuxième champ (`IDC_EDIT_NB2`), ainsi qu'un bouton pour sommer :

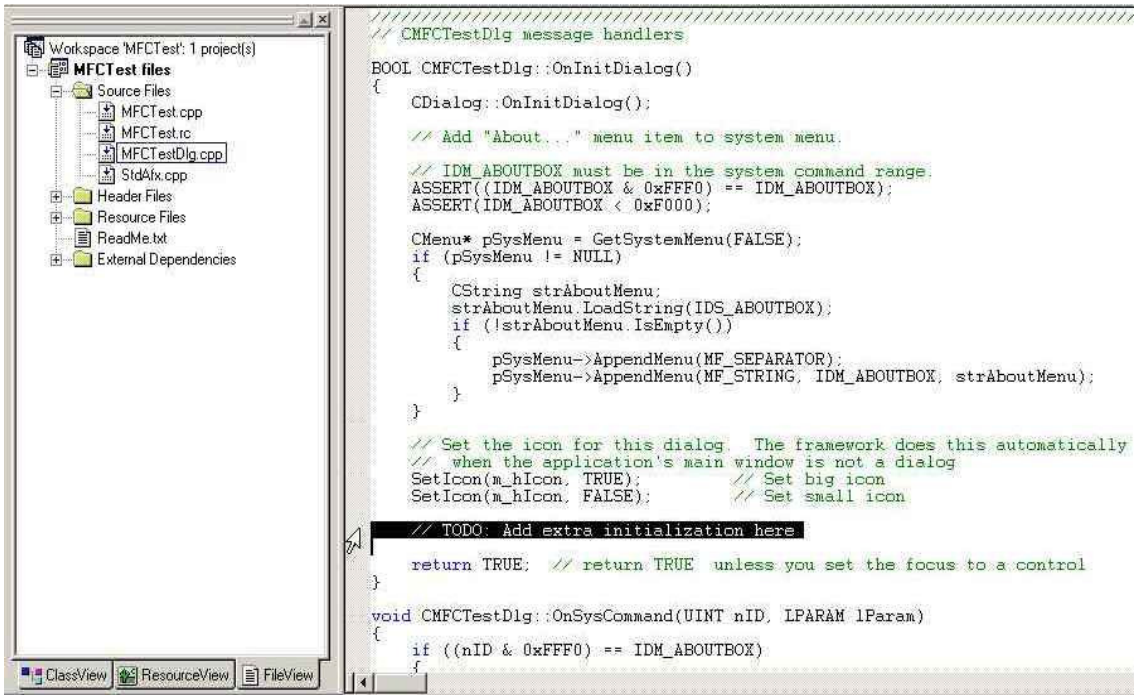


4.3 Quelques notes avant de commencer à programmer

Une erreur bête peut être évitée en vérifiant que *tous* les fichiers incluent le fichier stdafx.h, si jamais vous deviez en ajouter un.



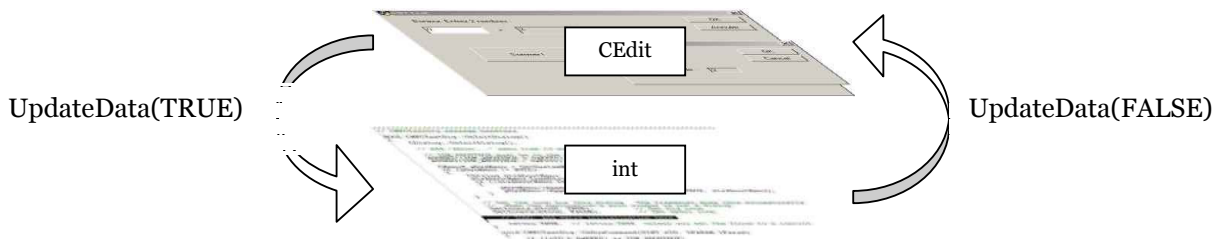
Le commentaire généré est assez (!) clair. En général il est indiqué où insérer votre code par un `//TODO` : …



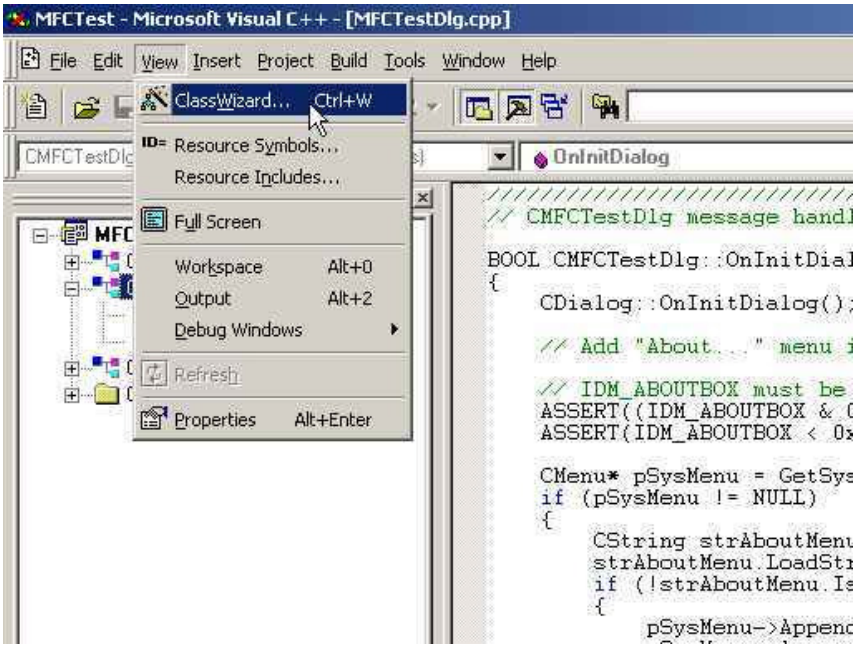
Avant de commencer, il va falloir initialiser les champs de l'interface graphique. En effet, la valeur d'un élément graphique est divisée en 2 variables :

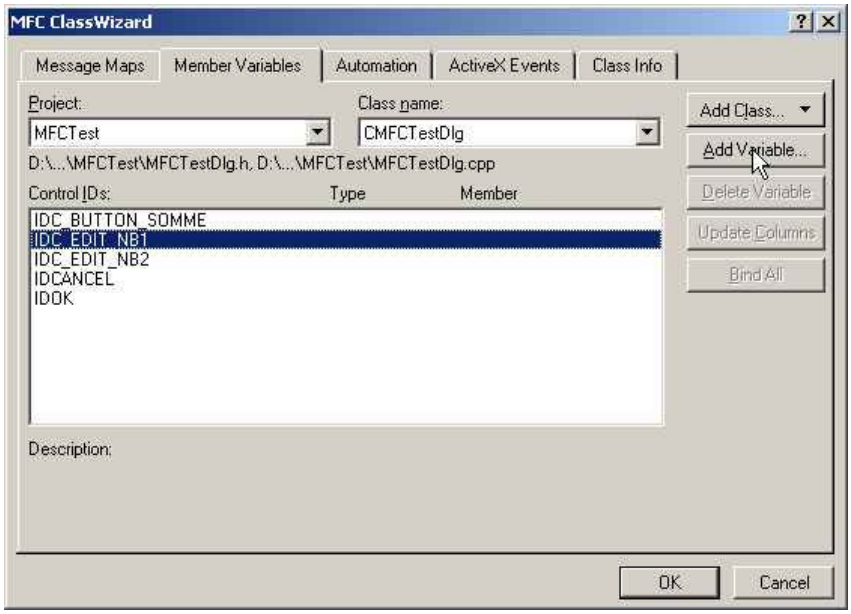
- une variable de type `CEdit` (car nous manipulons un champ) qui définit ce qui va être affiché dans l'interface
- une variable de type `int` qui va contenir l'opérande à manipuler

En effet, on ne peut pas manipuler directement le contenu du champ. On est obligé de le mettre à jour à partir d'une seconde variable selon le schéma suivant (sur lequel nous reviendrons) :

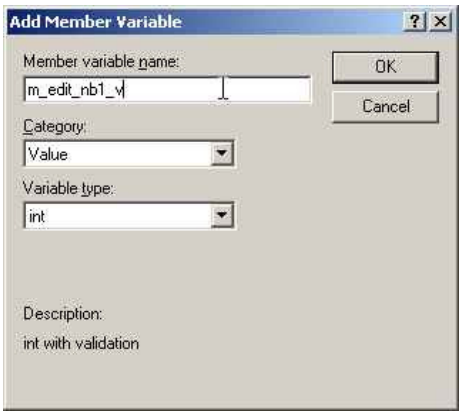


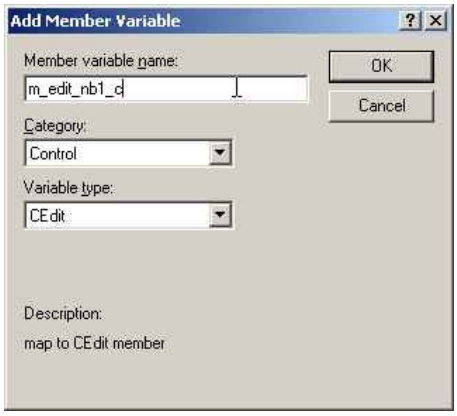
Ainsi, il va falloir définir des variables associées aux 2 boutons :





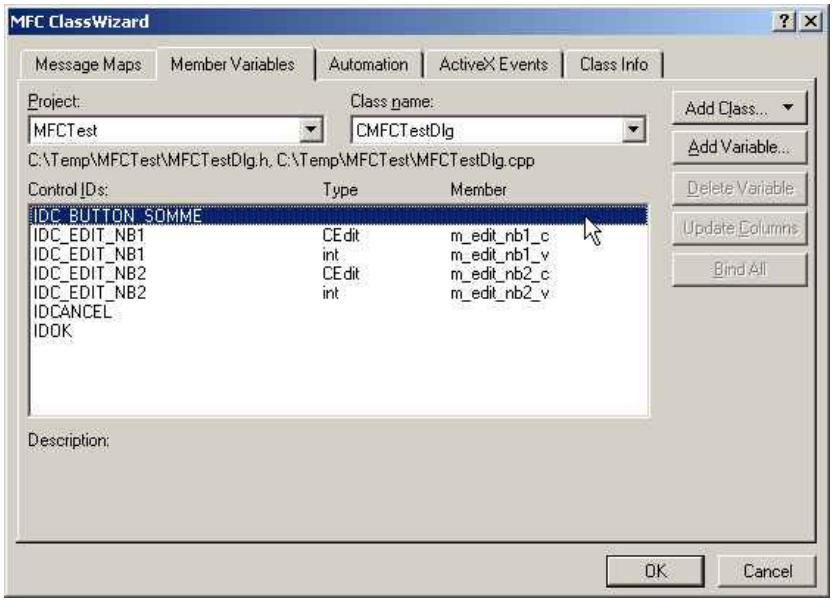
Ajouter 2 variables :





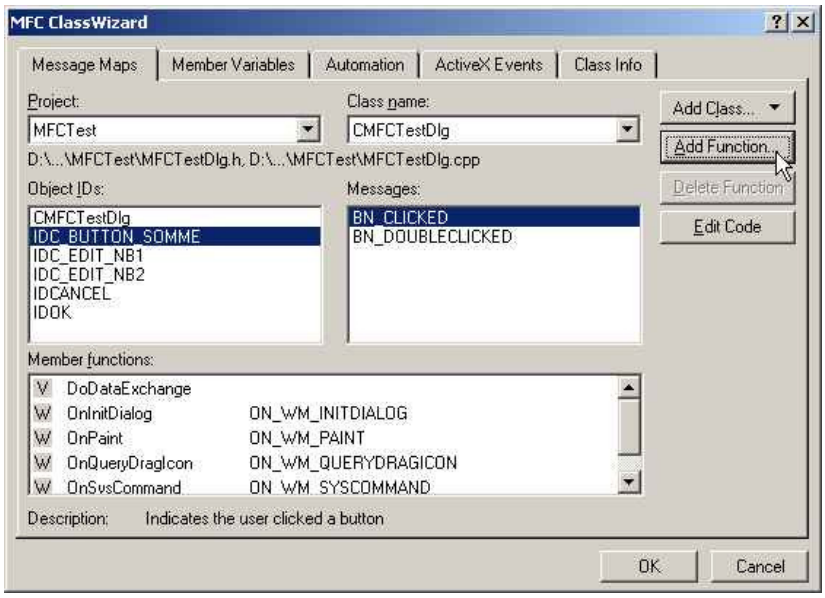
(ainsi, `UpdateData(TRUE)` copie les variables de types `Control` dans les variables de types `Value`)

Jusqu'à obtenir :

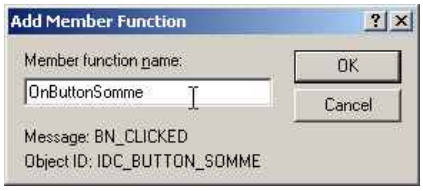


Ajoutons une méthode appelée lors de l'appuis sur le bouton somme. Retourner sur l'interface graphique :

cliquer sur le bouton somme et aller dans Edit➔Classwizard :

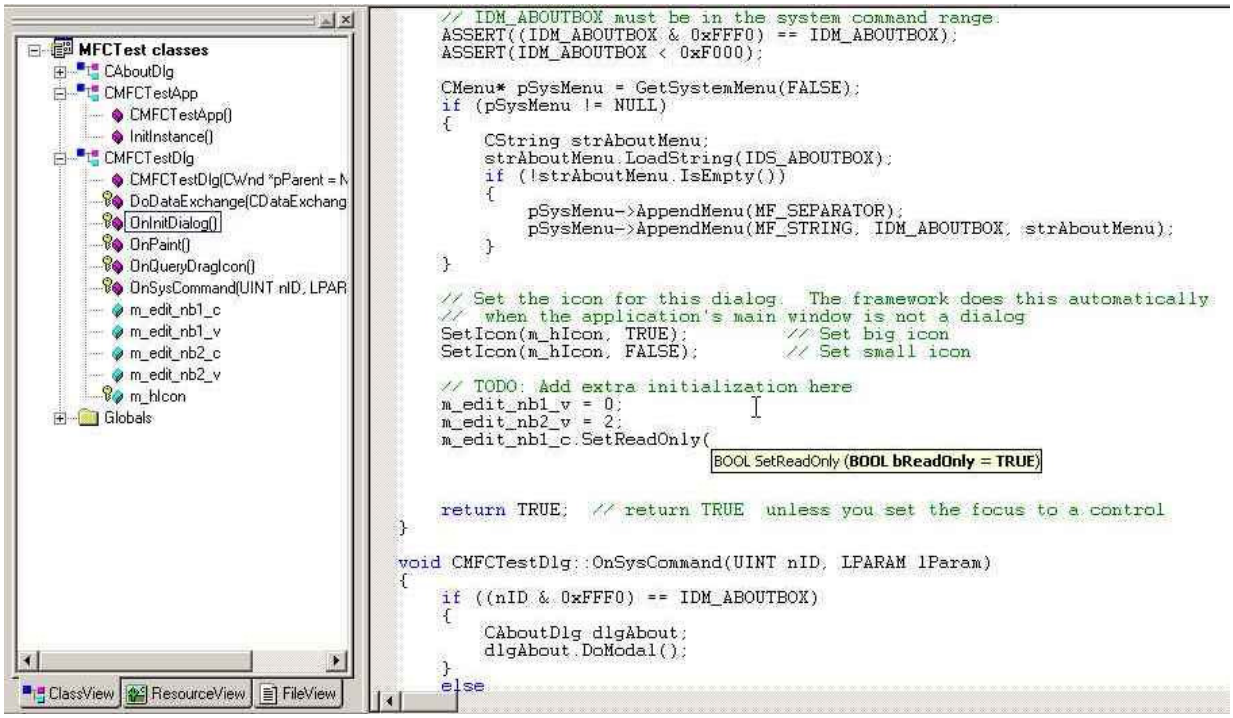


sélectionner le bouton somme et la méthode d'appel (ici un simple clic). Faire « Add function » :



Cette méthode s'appellera `OnButtonSomme`. Elle est automatiquement insérée là où il faut (dans le `.cpp` et le `.h`).

il va falloir définir les valeurs d'initialisation pour les champs :

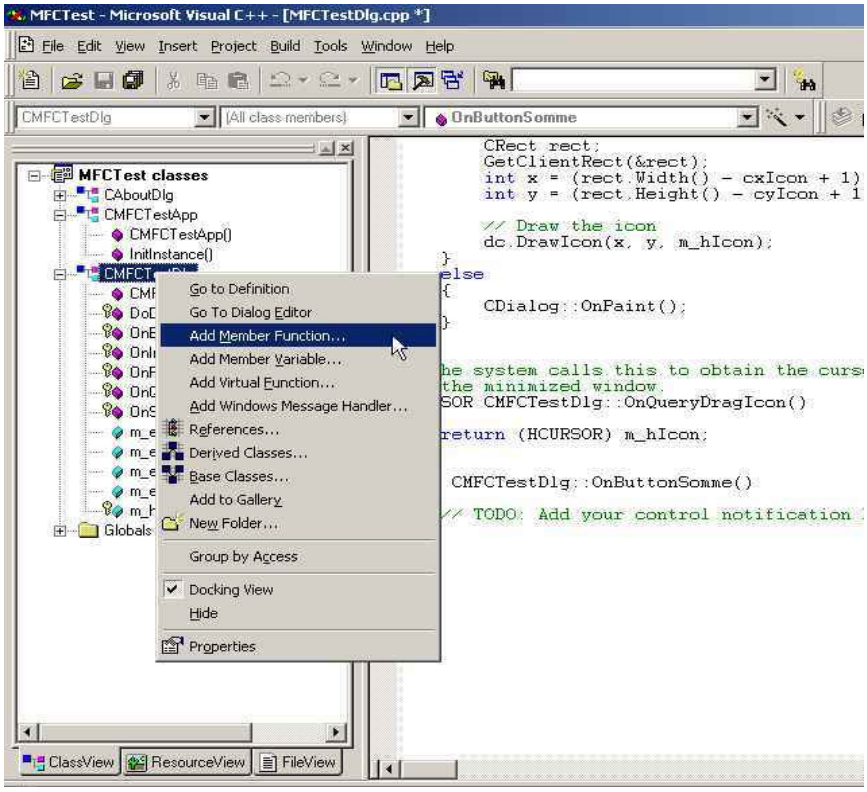


Le code est le suivant :

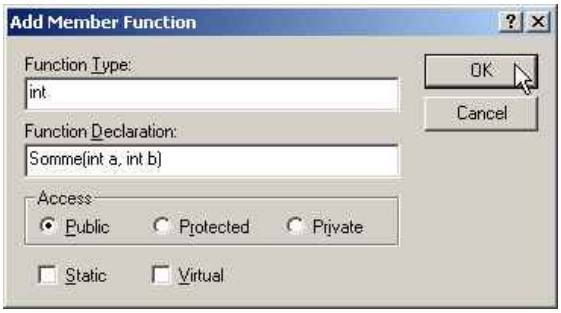
```
m_edit_nb1_v = 0;                    // on définit la première valeur à 0
m_edit_nb2_v = 2;                    // on définit la deuxième valeur à 2
UpdateData(FALSE);                   // on fait « remonter » les valeurs dans
                                        les champs de l'interface graphique
```

4.4 Passons à la programmation de notre méthode `Somme()`

Première étape : ajouter notre méthode à la classe `CMFCTestDlg`



elle prend 2 variables de type `int` renvoie un `int`



finalement, cliquer sur le bouton « Sommer ! » appelle `OnButtonSomme()` puis `Somme(int , int)` de la classe `CMFCTestDlg`.

La méthode `Somme()` contient le code suivant :

```
int CMFCTestDlg::Somme(int a, int b)
{
      return(a+b);
}
```

Corsons un peu le tout nous allons voir 2 méthodes pour afficher le résultat :

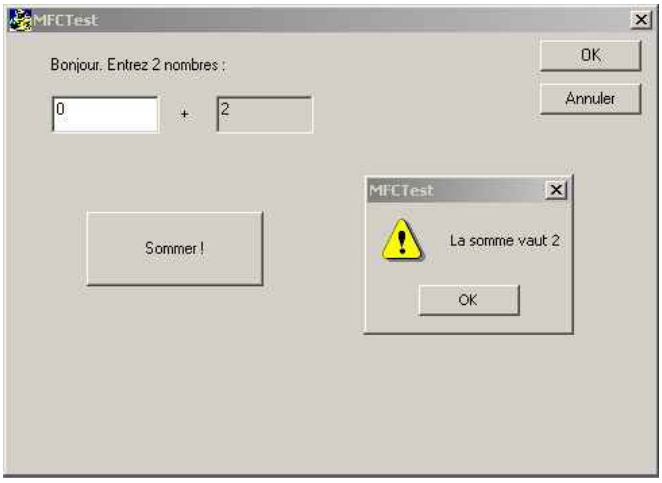
- un affichage par une fenêtre popup simple
- un affichage par une fenêtre ayant un champ grisé

4.4.1 Un popup simple

Le code est :

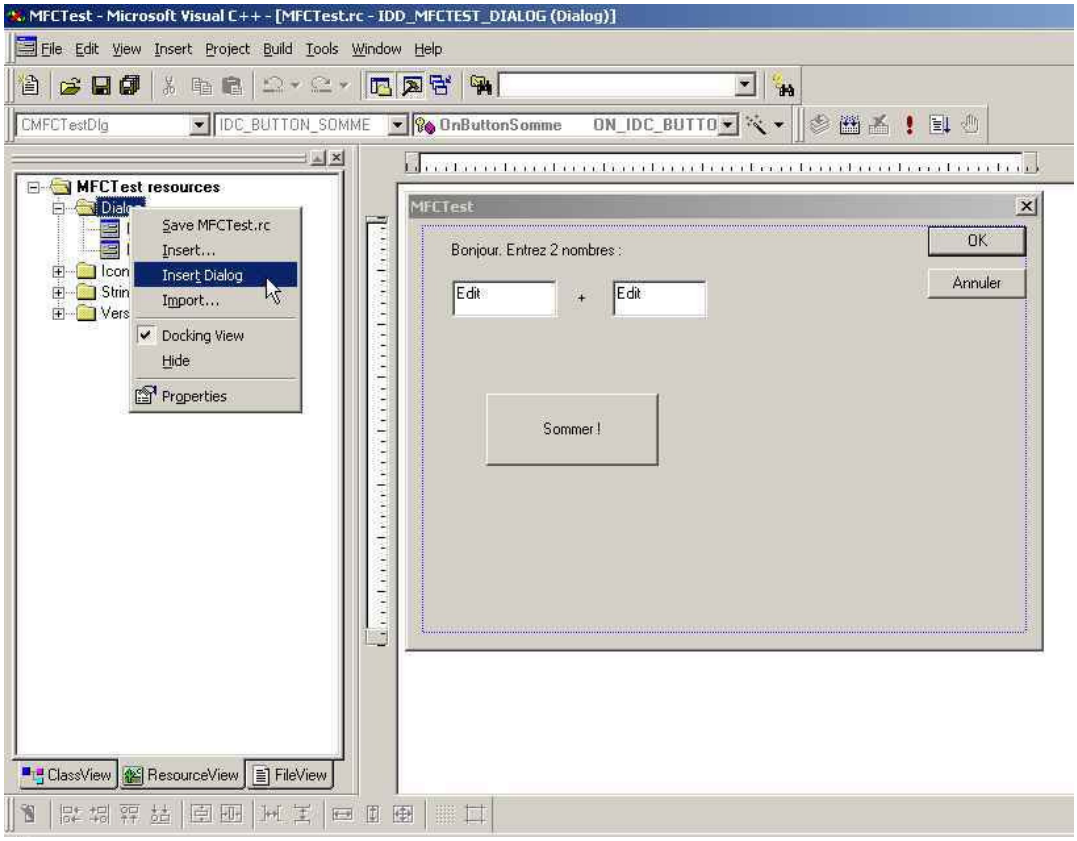
```
void CMFCTestDlg::OnButtonSomme()
{
   // TODO: Add your control notification handler code here

   int res;          // le résultat du calcul
   CString Str_res;  // la valeur de la chaine de résultat

   // On va chercher la valeur entrée dans les champs
   // (on copie m_edit_nbX_c dans m_edit_nbX_v)
   UpdateData(TRUE);

   // on fait la somme des valeurs
   res = Somme(m_edit_nb1_v,m_edit_nb2_v);

   // on forme la chaîne contenue dans le popup
   Str_res.Format("La somme vaut %d",res);

   // on affiche le popup
   AfxMessageBox(Str_res);
}
```

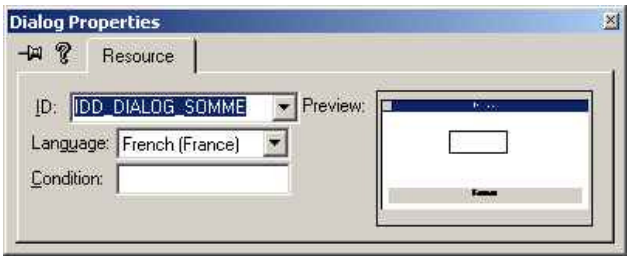
qui affiche :

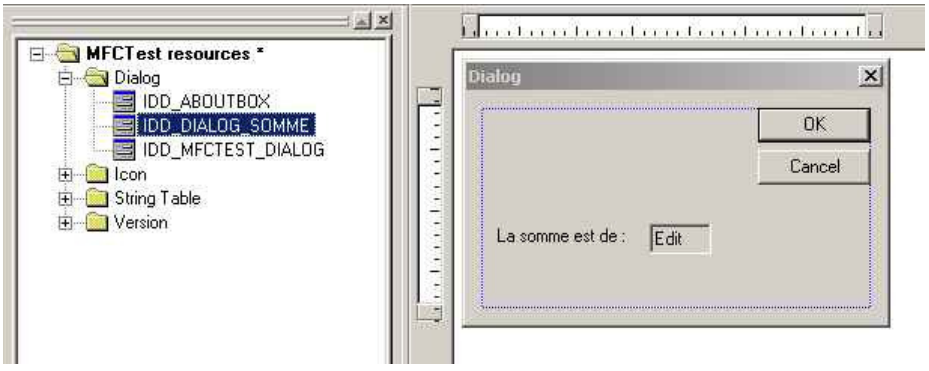


4.4.2 Une fenêtre avec un champ

Nous allons déjà créer la boîte de dialogue qui va apparaître lors de l'appui sur somme.



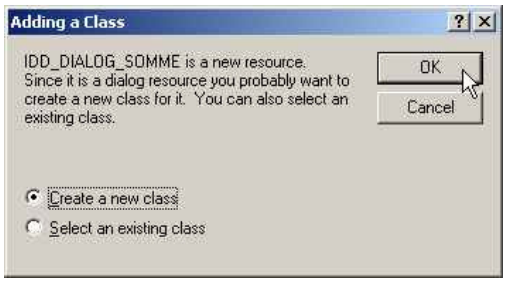


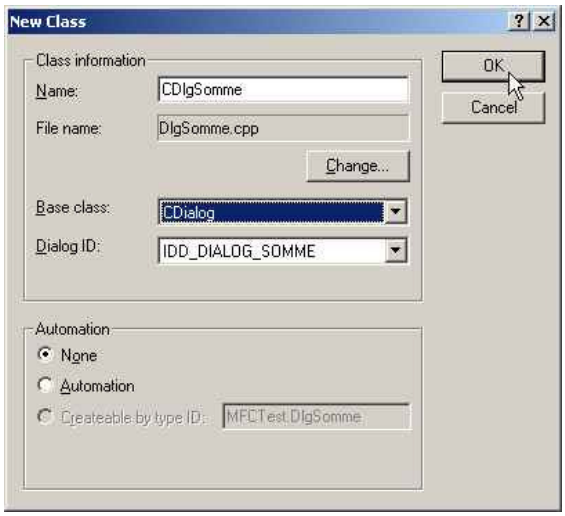


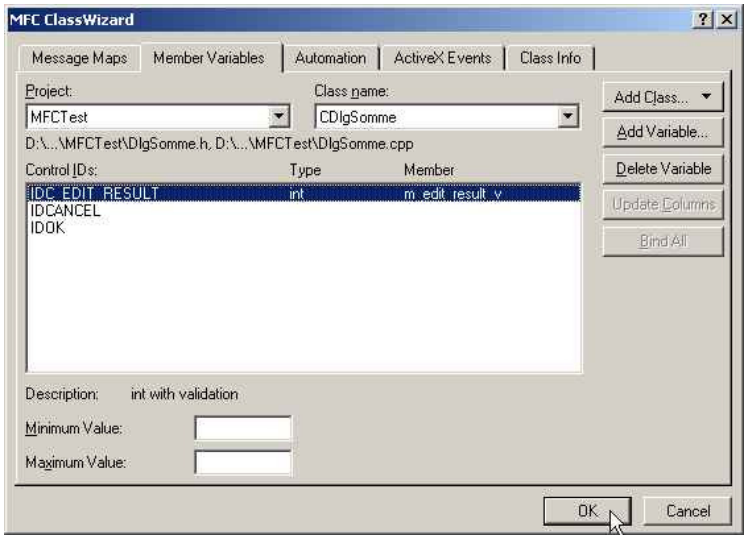
Ne pas oublier de changer le nom du champ :

Cette fois, nous n'ajouterons que la variable de type `Value` ! En effet, s'agissant d'un champ de résultat, l'utilisateur n'aura pas à entrer un chiffre, donc nous n'auront pas besoin de faire un `UpdateData(TRUE)`.

Allez dans le classwizard. Vous allez automatiquement générer la classe C++ liée à la boîte de dialogue nouvellement créée :







Une nouvelle classe (un fichier `DlgSomme.cpp` et un fichier `DlgSomme.h`) ont été créés. Allons jeter un coup d'œil dedans.

Les méthodes qui nous intéressent sont le constructeur et `OnInitDialog()`.

Pour pouvoir afficher le résultat dans le champ, il va falloir transmettre le résultat à la boîte de dialogue. Nous pouvons par exemple modifier le constructeur

```
CDlgSomme::CDlgSomme(CWnd* pParent /*=NULL*/)
     :CDialog(CDlgSomme::IDD, pParent)
{
   //{{AFX_DATA_INIT(CDlgSomme)
   m_edit_result_v = 0;
   //}}AFX_DATA_INIT
}
```

qui devient

```
CDlgSomme::CDlgSomme(CWnd* pParent, int insertres)
     :CDialog(CDlgSomme::IDD, pParent)
{
   //{{AFX_DATA_INIT(CDlgSomme)
   m_edit_result_v = insertres;
   //}}AFX_DATA_INIT
}
```

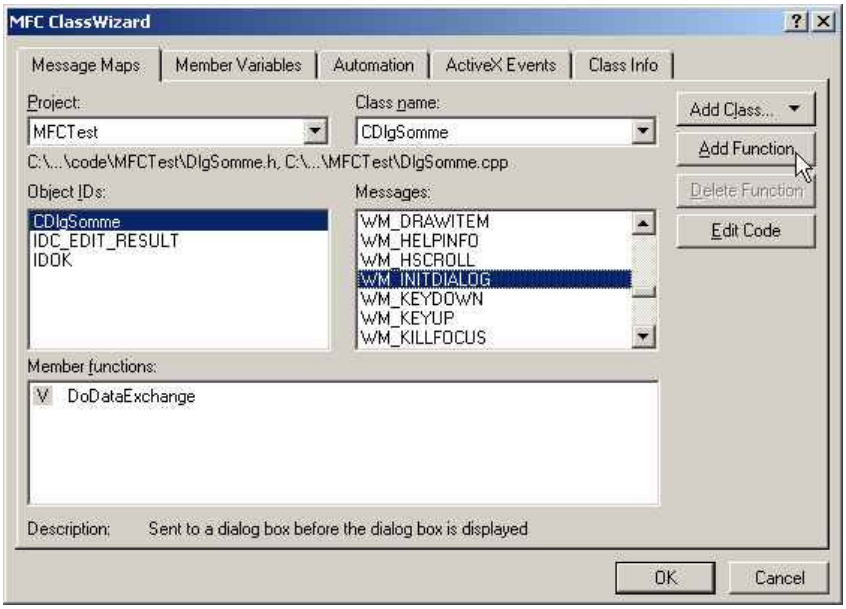
Ne pas oublier de modifier `DlgSomme.h` !

```
class CDlgSomme : public CDialog
{
// Construction
public:
   CDlgSomme(CWnd* pParent = NULL);   // standard constructor
...
```

devient

```
class CDlgSomme : public CDialog
{
// Construction
public:
   CDlgSomme(CWnd* pParent = NULL, int insertres = 0);  // standard constructor
...
```

Ajoutons une fonction `OnInitDialog()` à la classe `CDlgSomme` :



Modifier ensuite `OnInitDialog()` (qui est dans la classe `CDlgSomme`, je le rappelle) :

```
   // TODO: Add extra initialization here
   UpdateData(TRUE);
```

Il faut que la boîte de résultat apparaisse lors du clic sur le bouton Sommer ! Nous allons alors devoir ajouter l'appel à la fonction `OnButtonSomme()` de la classe `CMFCTestDlg`.
Il faut avant tout inclure dans `DlgSomme.cpp` :

```
   #include "DlgSomme.h"
```

puis modifier la méthode `OnButtonSomme()` :

```
   void CMFCTestDlg::OnButtonSomme()
   {
      // TODO: Add your control notification handler code here

      int res;            // le résultat du calcul
      CString Str_res;    // la valeur de la chaine de résultat

      // On va chercher la valeur entrée dans les champs
      // (on copie m_edit_nbX_c dans m_edit_nbX_v)
      UpdateData(TRUE);

      // on fait la somme des valeurs
      res = Somme(m_edit_nb1_v,m_edit_nb2_v);

      // on instancie une boîte de dialogue...
      CDlgSomme dlg(NULL, res);
      // ... que l'on rend visible.
      dlg.DoModal();
   }
```

5 Annexe

5.1 Réduire la taille de l'exécutable compilé

Par défaut, Visual C compile en mode debug. Pour notre exemple, l'exécutable généré pèse près de 1.4 Mo. Pour le réduire, il faut passer en mode release : dans le menu Build, sélectionner Set Active Configuration et choisir le mode release. La taille de votre exécutable passe alors à 120Ko environ.

5.2 Librairies statiques et dynamiques

Par défaut, et pour ce tutorial notamment, j'ai choisi d'utiliser des librairies statiques. Dans le cadre de librairies dynamiques, la taille de l'exécutable passe à 105Ko (mode debug), mais impose la présence dans le PATH des librairies nécessaires. A vous de voir.

	Statique	Dynamique
Debug	1400 Ko	120 Ko
Release	105 Ko	20 Ko

5.3 Analyse de l'exécutable par Scanbin

Nota : scanbin est un utilitaire de Jean-Claude Bellamy (www.bellamyjc.net).

L'exécutable a été compilé en mode release avec des librairies dynamiques.

```
DLL utilisées : MFCTest.exe
=========================================================

Appels directs
-------------------------------------------------------------

c:\winnt\system32 (4 DLL)
 dll-32 kernel32.dll 16/12/1999  779 536 octet(s) (V.5.00.2191.1 DLL du client API BASE Windows NT)
 dll-32 mfc42.dll    16/12/1999  995 383 octet(s) (V.6.00.8665.0 MFCDLL Shared Library - Retail Version)
 dll-32 msvcrt.dll   16/12/1999  295 000 octet(s) (V.6.10.8637.0 Microsoft (R) C Runtime Library)
 dll-32 user32.dll   16/12/1999  403 216 octet(s) (V.5.00.2180.1 DLL client de l'API Utilisateur de
                                                   Windows 2000)

Appels indirects
-------------------------------------------------------------

c:\winnt\system32 (2 DLL)
 dll-32 gdi32.dll    16/12/1999  234 256 octet(s) (V.5.00.2180.1 GDI Client DLL)
 dll-32 ntdll.dll    16/12/1999  502 544 octet(s) (V.5.00.2163.1 DLL Couche NT)
```