Method Maker for VB4

Introduction

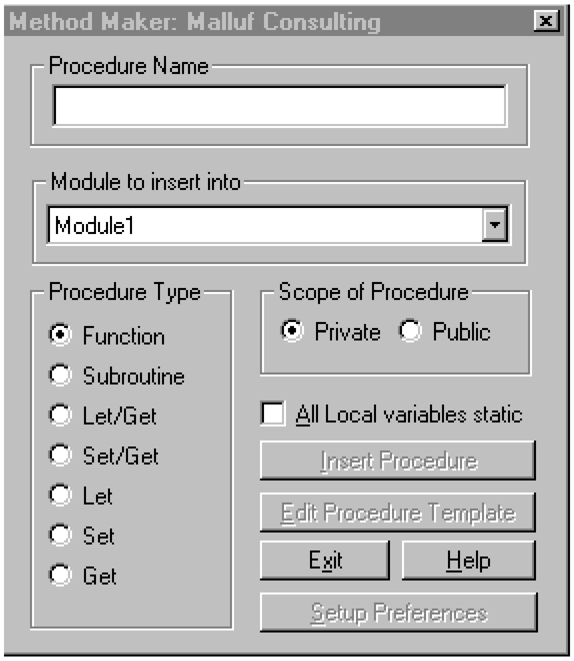
The Method Maker add-in is a small useful utility that provides the programmer a more complete set of procedure insertion options.
In this evaluation version, Method Maker provides you with the ability to insert predesigned procedure templates complete with a general error control block. The error control block can be edited to fit whatever error handling system you already have in place.

Procedures

Insert Procedures text here

Reference

Installing Method Maker



Module to insert into: Module1

Procedure Type: [✓ Function] Subroutine Let/Get Set/Get Let Set Get

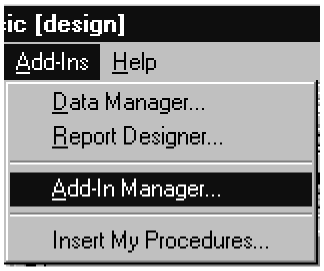
Scope of Procedure: [✓ Private] Public

☐ All Local variables static

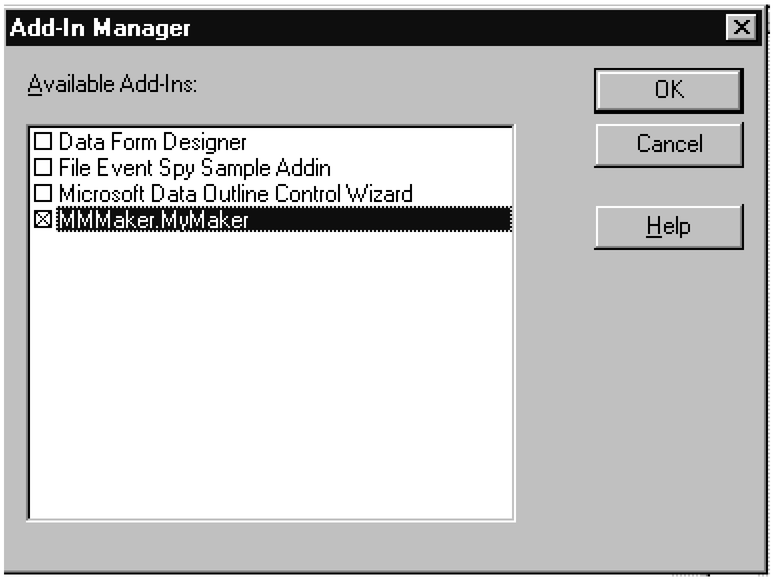
To install Method Maker, first create a directory for containing all of Method maker's files.
Unzip the MMAKE.ZIP file into the directory and from either Explorer or File Manager, doublclick the MMMaker.EXE file to run it.
It will show a license splash screen. close it from the form's control menu and the instalation will be complete.
The program will now be entered in your system registry and be made available to your VB4 Add-In Manger.

That's all there is to installing it!

Add-In Manager



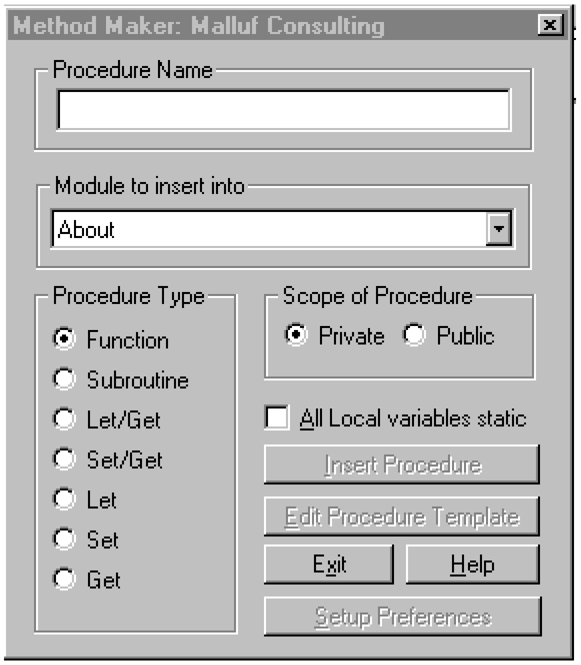
To make Method Maker available to your VB4 development environment, select the Add-ins menu item, then the Add-In Manager. You'll note in the illustration the 'Insert My Procedures" entry in the menu illustration. This is the Method Maker menu item that appears in the Add-ins menu after you have selected Method Maker through the Add-ins manager.



☐ Data Form Designer
☐ File Event Spy Sample Addin
☐ Microsoft Data Outline Control Wizard
☒ MMMaker.MyMaker

The Add-ins manager window is displayed here. it shows the MMMaker.MyMaker entry selected. I know, its a real original name isn't it.

Using Method Maker



Module to insert into: About

Procedure Type: [✓ Function] Subroutine Let/Get Set/Get Let Set Get

Scope of Procedure: [✓ Private] Public

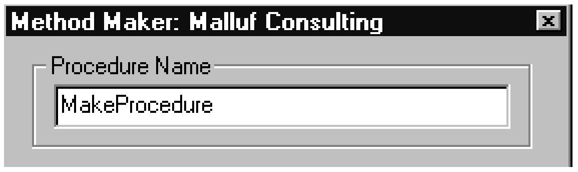
All Local variables static: ☐

The Method Maker form displayed on the left here provides you with several options for inserting procedures in your code modules.

You'll find that the options normally part of the VB4 IDE are there as well as some additional enhancements.

The Method Maker will insert your procedure much the same way that the standard IDE method will with the exception that it will provide your procedure with a standard template that includes a single entry/exit design and an error handler block that can be customized.

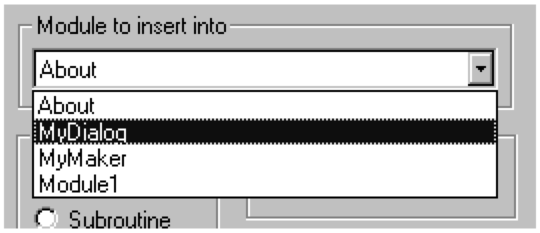
Procedure Types, Scope
Procedure Name
Inserting into a Module
The Procedure Template
Setup Preferences



Enter a valid name for the procedure in the Procedure Name window. As you enter a name the Insert Procedure command button will be enabled. If you delete the name from the Procedure Name text bos the Insert Procedure button will be disabled.

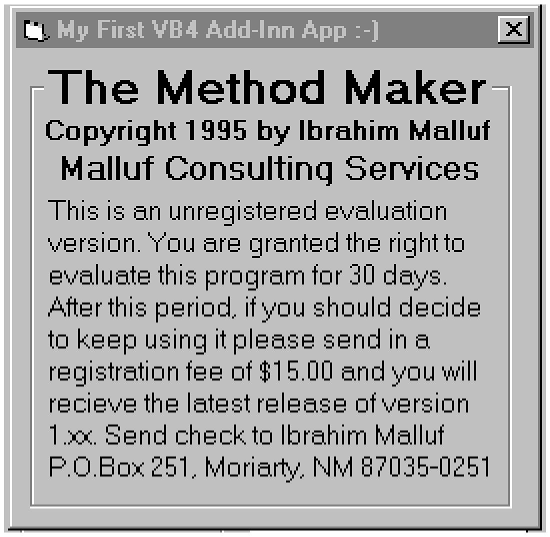
Try to use a name for you procedure that desribes the procedure's purpose as best you can. An example of this would be: MakeFunction or MakeLetGet. Both of these functions actually exist in this program. Note that you can easily tell what they do.

Insert Into Module



From this drop down box you can select the Form, Class, or Bas module to insert the procedure into.
It doesn't have to be the active module, though it will default to the active one when first loaded

Method Maker V1.xx License



My First VB4 Add-Inn App :-)

The Method Maker

Copyright 1995 by Ibrahim Malluf

Malluf Consulting Services

This is an unregistered evaluation version. You are granted the right to evaluate this program for 30 days. After this period, if you should decide to keep using it please send in a registration fee of $15.00 and you will recieve the latest release of version 1.xx. Send check to Ibrahim Malluf P.O.Box 251, Moriarty, NM 87035-0251

The Method Maker V1.00 is licensed to you for a period of 30 days to evaluate. After this period you must either send in the appropriate registration fees for each user in your organization or cease using it.

Registering with your $15.00 (real cheap) will get you a fully functioning version of this utility along with whatever enhancements I make to V1xx.

You'll also help me pay for my coffee habit :-)

Please Send Check or Money Order   for $15.00 for each copy to:

Ibrahim Malluf

P.O.Box 251

Moriarty, New Mexico 87035-0251

About Malluf Consulting Services

Malluf Consulting Services

Malluf Consulting services is located In Moriarty, New Mexico.
We specialize in Microsoft Windows development using the following tools:

Visual Basic 4.0 Enterprise Edition by Microsoft
SQL Server 6.0 Client/Server Database Development System by Microsoft
Microsoft Office Professional 95
Microsoft Back Office Products

We develop for all of the Microsoft platforms listed below. We create applications to your specifications to perform across any network supported by these platforms as well as on the Internet meeting the most stringent of requirements

- Microsoft Windows 3.11 & Microsoft Windows For Workgroups 3.11
- Microsoft Windows 95
- Microsoft Windows NT 3.51 Server and Work Station

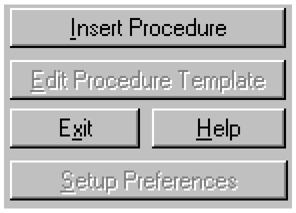
Index

#
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Glossary

≘
#
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

Command Buttons



As you can see there are five command buttons. Three of them are available in the evaluation version you now have. The other two will be available with the registered version. Hey, what do you want for nuthin! A lot of you will probably just use this tool as is without registering.
I got to have some kind of incentive, right?

Insert Procedure
Edit Procedure Template
Exit
Help
Setup Preferences

Insert Procedure

This command becomes available when the Procedure Name text box has more than two characters in it. Why two characters? Just because!

When you select this command the selected procedure(s) are inserted into the specified code container.

Edit Procedure Templates

Not until you send me that $15.00!

When you do send it, I'll send you the version that allows you to create your own brand of standard procedure templates. You'll be able to include whatever header information you want, as well as other structures that your particular needs require.

Method Maker Help

Geeeeze Louize, you're already using it. (stole the Geeeze Louize from another programmer)

Setup Preferences

This Setup Prefere's registered users
So I nag alot.   I could use the 15 bucks and you can use this program. so send it.

- Choose Default Values
- Toggle always on top property
- Default procedure headers
- Send in suggestions and I might add them here!

Function Procedures

```
Private Function MakeFunction()
        'error call
        On Error GoTo BadMakeFunction

'single exit block
ExitMakeFunction:

        Exit Function
'error routine block
BadMakeFunction:
        'Show error in message box if the
        'show error preference is true
        If colPreferences("ShowError") Then
                MsgBox Error
        End If
        'set function to false
                MakeFunction = False
        Resume ExitMakeFunction
End Function
```

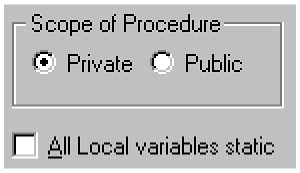
This is an example of the function procedure template.
two jump labels are created, one for the exit block and the other for the error block.
The error block can be modified to fit your purposes through a text file called ErrBody.Txt
Future versions will allow you to fully cutomize the body of each type of procedure.

The above structure represents my prefered method of programming. Both a sucessful and failed execution should always exit at the same place. This way any procedure cleanup that is needed gets done regardless and no code need be duplicated. Future versions will allow you to apply your own coding philosophy to this structure. For now, only the error block is custumizable.

Method Maker Exit

See, I'm really a nice guy and not completely mercenary. I left a way for you to exit the program without charging extra for it

Procedure Scope


Scope of Procedure

(● Private) ○ Public

☐ All Local variables static

The default scope of procedures when using Method Maker is Private. This is the opposite of the VB-IDE which defaults to public. I guess its a matter of opinion, and my opinion is that the default should always be Private. If you want the default to be Public then register Method Maker and you'll be able to set your own defaults.

Anyway, you can select Public if you want the procedure to be public. You can also select the All Local Variables Static box to declare the procedures local variables automatically static.

Procedure Types

Visual Basic provides three types of procedures. They are the Function, Subroutine, and Property procedures. They all have their particular applicability with a certain amount of overlap. The Function and Subroutine procedures have been part of Visual Basic since the beginning.   The Property Procedures on the other hand are entirely new to Visual Basic being introduced with VB 4.0

Function Procedures
Subroutine Procedures
Property Procedures

Subroutine Procedures

```
Private Sub FindComponents()
        'set up jump to error block
        On Error GoTo BadFindComponents
'exit block
ExitFindComponents:

        Exit Sub
'error block
BadFindComponents:
        'Show error in message box if the
        'show error preference is true
        If colPreferences("ShowError") Then
                MsgBox Error
        End If
        Resume ExitFindComponents
End Sub
```

This is an example of the subroutine procedure template.
Two jump labels are created, one for the exit block and the other for the error block.
The error block can be modified to fit your purposes through a text file called ErrBody.Txt

Future versions will allow you to fully cutomize the body of each type of procedure.

The above structure represents my prefered method of programming. Both a sucessful and failed execution should always exit at the same place. This way any procedure cleanup that is needed gets done regardless and no code need be duplicated. Future versions will allow you to apply your own coding philosophy to this structure. For now, only the error block is custumizable.

Property Procedures: Let/Set/Get

The Property Procedures are new in Visual Basic 4.0. They provide a secure way of exposing your object values. They directly replace Global variables in a dramatic fashion. Now you can control the changing of any value in your application in one place complete with validation rules.

There are three Property Procedures. They are:

Property Let Procedure
Property Set Procedure
Property Get Procedure
The Let and Set Property Procedures allow the changing of a value within the rules laid out within the procedure.
The Get Property Procedure allows the Reading of a Value within the rules laid out within the procedure.

In order to make an object value a read/write value you must provide a matching pair of properties such as Let/Get or Set/Get. You will find that the Method Maker gives you the choice of making these matched sets, or of creating a write-only or read-only property.   The standard VB IDE automatically provides matched pairs only. To have a read-only or write-only with the standard IDE you must delete one of the procedures

Property Let Procedure

```
Private Property Let Selection(vNewValue)
        'set up jump to error block
        On Error GoTo BadSelection
'exit block
ExitSelection:

        Exit Property
'error block
BadSelection:
        'Show error in message box if the
        'show error preference is true=
        If colPreferences("ShowError") Then
                MsgBox Error
        End If
        Resume ExitSelection
End Property
```

The Property Let Procedure provides a way for you to modify the value of an object's internal variables while maintaining full control of validating the changes.

You can also initiate any kind of action when this property value is affected.

The above structure represents my prefered method of programming. Both a sucessful and failed execution should always exit at the same place. This way any procedure cleanup that is needed gets done regardless and no code need be duplicated. Future versions will allow you to apply your own coding philosophy to this structure. For now, only the error block is custumizable.

Property Set Procedure

```
Private Property Set Preferences(vNewValue)
        'set up jump to error block
        On Error GoTo BadPreferences
'exit block
ExitPreferences:

        Exit Property
'error block
BadPreferences:
        'Show error in message box if the
        'show error preference is true=
        If colPreferences("ShowError") Then
                MsgBox Error
        End If
        Resume ExitPreferences
End Property
```

The Property Set procedure is used to pass an Object reference to a Form, Class, or Bas module.
Two jump labels are created, one for the exit block and the other for the error block.
The error block can be modified to fit your purposes through a text file called ErrBody.Txt

The above structure represents my prefered method of programming. Both a sucessful and failed execution should always exit at the same place. This way any procedure cleanup that is needed gets done regardless and no code need be duplicated. Future versions will allow you to apply your own coding philosophy to this structure. For now, only the error block is custumizable.

Property Get Procedure

```
Private Property Get Preferences()
        'set up jump to error block
        On Error GoTo BadPreferences
'exit block
ExitPreferences:

        Exit Property
'error block
BadPreferences:
        'Show error in message box if the
        'show error preference is true
        If colPreferences("ShowError") Then
                MsgBox Error
        End If
        Resume ExitPreferences
End Property
```

The Property Get Procedure provides the read part of an object property. It must be declared as a type that matches the value it exposes. If it is a string then you declare it As String, if it is a collection then you declare it As Collection, and so on.

The above structure represents my prefered method of programming. Both a sucessful and failed execution should always exit at the same place. This way any   procedure cleanup that is needed gets done regardless and no code need be duplicated. Future versions will allow you to apply your own coding philosophy to this structure. For now, only the error block is custumizable.