ClassMaster

What is ClassMaster?

Visual Basic 4.0 now supports a programming concept taken from the pages of object-oriented development. VB developers now have the opportunity to experiment with the concepts of *classes*. Classing allows developers to define applications in terms of objects, their characteristics, and the actions that can be performed upon them as a single unit that can be reused among different applications.

ClassMaster is a Visual Basic 4.0 16/32 add-in that allows a developer to define classes, create member variables, establish naming conventions, and then automatically generate .cls module that includes all member variable declarations as well as a set of basic Property Let/Set/Get procedures that establish the class interface. ClassMaster can then insert the constructed class into the current Visual Basic project.

VB ClassMaster Help: Overview

Visual Basic Class Structure

Visual Basic classes consist of member variables, which themselves can be declared Public or Private. Public member variables can be accessed in any part of a program that uses the class; private members can be accessed only within SUB's or Functions declared within the Class.

In a class, special procedures can be written to access private member variables, called *Property procedures*. Three types of property procedures are available; Property Set, Property Get, and Property Let.

If a class developer declares all member variables to be Private, and then writes Property procedures to access them, developers who use the class in their applications need only know about the names of the procedures, and not how the procedures were written. That embodies an object-oriented design concept called encapsulation.

ClassMaster: Class Generation

Class Building Options

Although VB enforces no particular standards regarding the naming of classes, member variables, or procedures, it is generally a good practice to incorporate some kind of notation standard within the classes you define.

As an example, member variables can be prefixed with a string such that any reference to it in code will clearly be indicated by the member prefix. One standard used in C++ is to prefix member variables with "m_"; this is the default member variable prefix in ClassMaster.

The name of each member should also be prefixed with a notation that indicates the type of data it represents. As an example, 'iNumber' indicates an Integer, 'sName' indicates a String. ClassMaster supports this simple naming convention.

The names for Property procedures should include the name of the object, exclusive of prefixes, and a verb that indicates the "direction" of the data. By default, Let, Get, and Set are the default verbs for each of the three property procedure types. The ordering of the names and verbs for each Property Procedure can be defined within ClassMaster. This means that, for a member named Address, the Property Let procedure could be named AddressLet or LetAddress, assuming Let was chosen as the default verb for the Property Let procedure.

ClassMaster: Opening menu

ClassMaster Main Screen

The ClassMaster Main Screen shows the name of the current class, or <none> if no class is presently defined. The member list shows the names of the member variables, exclusive of any prefixes.

The "Add Member" button allows a new member to be defined for the class; the "Options" button shows the ClassMaster Options dialog. The "Create Class" button causes ClassMaster to create the .cls file that holds the definition of your class as defined to ClassMaster.

ClassMaster Options

The ClassMaster Options dialog shows all of the class creation options available. The 'Naming Rules' section contains allows for the selection of options to use/not use a member variable prefix for each member variable. Clicking the 'Use Variable Type Prefixes' option causes the first letter of the member variable type to be inserted in front of the member name when the class is generated.

The Property Naming Options allows the designation of verbs that are to be used when ClassMaster builds the default Property Let/Set/Get procedures for each member of the class. The verbs precede the object names within the procedures if the 'Verb First' name ordering option is selected; names are first if the 'Name First' option is selected.

For example, if the Let verb is 'Let', and 'Name First' option is selected, the following code would be generated by ClassMaster:

```
Private m_sAddress as String

Public Property Let AddressLet(Param as String)
      m_sAddress=Param
End Property
```

Member Definition

Defining class members in this dialog requires the definition of the member's name, its scope, and its type. The name can be any valid VB variable name. Its scope means the member variable will be declared either Public or Private, and Read-Only determines whether a Property Let procedure will be created for the member. Checking "read-only" means no Property Let procedure will be generated for the member by ClassMaster.

The DataType dropdown list allows for the designation of data type for the member variable. At this time, only six types are supported by ClassMaster. This will be expanded in future versions. Presently, supported types include Integer, Long, Double, String, Variant, and Object.

Note that a member whose type is Object will have a Property Set procedure defined instead of a Property Let procedure.

'Don't use prefix on member variables

Click on 'Don't use prefix on member variables' to prevent any special designation being generated for member variables.

Click 'Use member prefix' and specify the prefix in the adjacent text box to be used when the member is generated in code.

Use Variable Type Prefixes in Member Names

Click on 'Use Variable Type Prefixes in Member Names' to instruct ClassMaster to prefix all member variables with a one-character prefix representing the type of variable the member represents; i.e. i for Integer, s for String, etc.

Property Procedures - An Overview

Ideally, if each member of your class is declared Private, the developers that use your class will need only to know the publicly delcared functions you have created that 'expose' the class to the application. The details of how your class works are of no interest to the developer; just the functions he or she can use to manipulate the class as they need for their application.

Property procedures are an important part of this concept. A Property Let procedure allows a value to be assigned to a private member variable; a Property Get procedure can retrieve a value from a private variable; a Property Set procedure can establish a reference to an object.

ClassMaster will build a Property Let and Property Get procedure for each member of your class. You may wish to edit these 'default' procedures to tailor their function to your specific needs. (Member variables declared as type 'Object' will be given a Property Set procedure instead of a Property Let.) For each member that is declared Read Only in the Member Definition dialog, no Property Let procedure will be generated by ClassMaster.

It's important to note that there is nothing special about the default code generated in each of these procedures. They represent only basic set of code that can be amended and tailored as you need for your particular class.

Property Procedure Verbs

When building the default set of Property Procedures for your class, ClassMaster will insert a verb indicating the 'direction' of data flow for each type of procedure. This verb, combined with the member name (exclusive of prefixes) represents the name of the property procedure. The default names for the Let, Get, and Set procedures are Let, Get, and Set, respectively. Each of the default names can be changed in the Class Construction Options dialog.

Name Ordering: Name First

When the Name First option is specified in the ClassMaster Options dialog, Property Procedure names will be built with the variable name first (exclusive of prefixes), followed by the verb assigned for each procedure type.

```
Public Property Let AddressLet(Param as String)
    m_sAddress=Param
End Property
```

or, w/Verb First set

```
Public Property Let LetAddress(Param as String)
      m_sAddress=Param
End Property
```

Name Ordering: Verb First

When the Verb First option is specified in the ClassMaster Options dialog, Property Procedure names will be built with the variable name following (exclusive of prefixes) the verb assigned for each procedure type.

```
Public Property Let LetAddress(Param as String)
    m_sAddress=Param
End Property
```

or, w/Name First set

```
Public Property Let AddressLet(Param as String)
      m_sAddress=Param
End Property
```

Member Name

The Member Name is the name by which the member is defined in ClassMaster. Depending upon the construction options selected, the name may be prefixed with either a member variable prefix (which applies to all member variables), and/or a variable type prefix, indicating the data type the variable represents.

'In code:'

The 'In Code' label within the Member definition dialog displays the way the current member variable will appear in the code generated by ClassMaster. The Member List in the main ClassMaster screen always depicts the members of the class by the member name without any prefixes.

Scoping: Private or Public

Members declared Private are accessible only to other functions and Sub procedures declared in the Class. Public members are available to the application that uses the class.

Scoping: Read Only

A member declared as Read Only will have no Property Let (or Set for members declared as Objects) created for them by ClassMaster.

Data Type list

The Data Type list allows the user to specify the data type of the current member variable.

About ClassMaster

David Whitney
9101 Southlake Drive
Oklahoma City, OK 73159

email all queries and bugreports to: intrepid@ionet.net

Thanks!