Learning VB.Net

Tutorial 18 – Classes Initialization and Finalization

Hello everyone… welcome to vb.net tutorials. These are going to be very basic tutorials about using the language to create simple applications, hope you enjoy it. If you have any notes about it, please send them to notes@mka-soft.com I will be happy to answer them. Finally if you find these tutorials are useful, it would be nice from you to send a small donation via PayPal to donation@mka-soft.com.

Tutorial posted on 2010-May-09.

Classes Initialization and Finalization

The previous tutorial showed how to create a class, and how to add methods, and attributes to it. Today we see how to initialize the objects using the New method. First open the previous class example “testclass”. Add a new class to the project, and call it: ContactList.

This class will be used to store the contact information in the array and manage it . In the class file add the following code:

```
    Dim ContactArr() As ContactInfo             ' the array of object, all elements points to nothing
    Dim C As Integer                            ' the number of objects in the array
```

These are used to store the contact information, and the number of elements in the array used. Next add the following method:

```
    Public Sub AddNewContact()
        C = C + 1                               ' the number of objects increases by one
        ContactArr(C - 1) = New ContactInfo     ' create the object
        ContactArr(C - 1).ReadContactInfo()     ' read the information
    End Sub
```

This one adds a new contact, then add:

```
    Public Sub RemoveContact(ByVal Name As String)
        ' search for the contact
        For I = 0 To C - 1
            If ContactArr(I).Name = Name Then


                ' next remove the contact from the array by shifting the other objects
                Dim J As Integer
                For J = I + 1 To 999
                    ContactArr(J - 1) = ContactArr(J)
                Next

                ' the number of elements reduces by one
                C = C - 1

                ' exit the block
                Exit Sub
            End If
        Next

    End Sub
```

Which will remove a contact based on name. Also, add the following method to fill the data grid view:

```
    Public Sub FillDGV(ByVal DGV As DataGridView)
        ' clear the data grid view
        DGV.Rows.Clear()

        Dim I As Integer

        ' loop over all the contacts
        For I = 0 To C - 1
            ' add contact information
            DGV.Rows.Add(ContactArr(I).Name, ContactArr(I).Address, ContactArr(I).Tel)
        Next

    End Sub
```

Now comes the constructor, write down the following:

```
    Public Sub New()
        ' first constructor, set the number of elements to zero, and set array size to 1000
        C = 0
        ReDim ContactArr(0 To 999)
    End Sub
```

The name of this method is: New, and by default, when the compiler sees this, it knows that this method should be called automatically as soon as the object is created. So basically this method tells the computer to set the value of the counter C to zero, and make the array capable of storing 1000 objects as soon as the ContactList object is created. To test this, Go to the form, and modify the code to be like this:

```
Public Class Form1

    Dim OBJ As ContactList


    Private Sub AddToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles AddToolStripMenuItem.Click

        OBJ.AddNewContact()
        OBJ.FillDGV(DGV)

    End Sub

    Private Sub RemoveToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RemoveToolStripMenuItem.Click

        ' check if no rows are selected, if so no need to execute further code, exit the subroutine
        If DGV.SelectedRows.Count = 0 Then
            Exit Sub
        End If

        Dim N As String

        ' get the selected name, it is the first column (cell zero)
        N = DGV.SelectedRows(0).Cells(0).Value

        OBJ.RemoveContact(N)
        OBJ.FillDGV(DGV)

    End Sub


    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load

        OBJ = New ContactList()
        OBJ.FillDGV(DGV)

    End Sub

End Class
```

Check out the code, and specifically the Form1_Load subroutine. When the line:

```
        OBJ = New ContactList()
```

is executed, the New subroutine is called directly. You don't have to do the call yourself, it is automatic. Also the constructor is executed only once.

So basically the constructor helps you prepare your object before using it. To make sure the constructor is being called, try to add a **MsgBox** call in the New method and see how it works.

You can actually create a number of different constructors, and later on you can choose which one to use based on the parameters you pass to it. For example, let us add another constructor to our class:

```
    Public Sub New(ByVal NoOfReads As Integer)
        ' second constructor, set number of elements to zero, and set array size to 1000
        C = 0
        ReDim ContactArr(0 To 999)

        ' add the contacts
        Dim I As Integer
        For I = 0 To NoOfReads - 1
            Me.AddNewContact()
        Next
    End Sub
```

This constructor allows you to read a number of contacts as soon as you initialize the object without the need of going to the menu and select add contact. In order to call it, simply use it like this:

```
        OBJ = New ContactList(3)
```

When the compiler sees the parameters (3), it searches for the constructor that accepts an integer as a parameter and calls it. You can create as many constructors as you need. The important thing is that the constructor name is always New, and each constructor should have different parameters (either in number or in data type to help the compiler distinguish them). Try the new constructor, and see how it works.

The last thing is the destructor. A destructor or finalizer is a method that is called when an object is destroyed, i.e. its resources are returned into memory. Try adding the code below:

```
    Protected Overrides Sub Finalize()
        ' this is how to terminate a class
        Dim I As Integer
        For I = 0 To C - 1
            ContactArr(I) = Nothing
        Next


        MyBase.Finalize()
    End Sub
```

Don't worry about the **MyBase**, or **Protected**, or the **Overrides** keywords for now, we will check these in later tutorials, but for now, just keep in mind that this one is being called when the object is destroyed. As you can see what we are doing here is we are looping over all the contactinfo objects and set them to nothing (which means we don't need them anymore, and we want them to be destroyed). Then after that we destroy the object. Try this code out, and see what happens when you place an MsgBox in this method.

So this is all for today. If you need the source file, you can get it from the web site. If you have notes about this tutorial, email me at: **notes@mka-soft.com**.

Thanks.

mkaatr