Learning VB.Net

Tutorial 16 – Modules

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 <u>notes@mka-soft.com</u> 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 <u>donation@mka-soft.com</u>.

Tutorial posted on 2010-March-27.

Modules

In vb.net you can write place your code in different files. Some of these files are called modules. A module is a file that contains vb code only (i.e. functions, structures, subroutines...). It does not include GUI like buttons, lists, menus ...etc. Now to add a module to your project select project then add module



After that you provide module name:

| <u>Categories:</u> | | Templates: | | |
|--|------------------|---|---|--|
| Common Item Code Data General Web Windows F Reporting Workflow WPF | s orms | Visual Studio installed templates Statution About Box Class Diagram COM Class Custom Control Dialog Interface LINQ to SQL Classes Local Database Cache MDI Parent Form Report Resources File Splash Screen User Control | Class Code File Component Class DataSet Explorer Form JScript File Local Database Login Form Module Report Wizard Service-based Database Text File User Control (WPF) | |
| A file for storing g | oups of function | ns | ~1 | |
| <u>l</u> ame: | Generalvb | | | |

| iolution E | xplorer - Solution 'TestSales' (1 project) 교민도 중 | * 1 |
|------------|---|----------------|
| | and Project My Project AddSale.vb Form1.vb | |

You can see the module file is added into your project. The code you see is:

```
Module General
End Module
```

Now you can write the functions and subroutines in this module. For example:

```
Module General

' define the sales structure

Public Structure SalesStruct

Dim Item As String

Dim Count As Integer

Dim Price As Single

Dim Total As Single

Dim TheDate As String

Dim Flg As Boolean

End Structure

End Module
```

So you might be wondering what difference does modules make in a program? Well modules helps you place related functions, subroutines, and other coding that you make in one place, so that it becomes easier for you to find, and easier for you to work with, and make it possible for other parts of your program to use the code.

To illustrate this consider that you have a two or three forms that require some sort operation. Instead of writing the code in one form which makes it part of that form, you place the code in a module, and sort function/subroutine becomes available to all the forms. Later on you can even take the code of the sort operation and add that to another project and you will find it works without modification (assuming the code is done correctly). Now the following example (the sales vb project included on the web site). This project contains one module:



The idea of this project is to add a number of items you were able to sale, and later on you could find the total of sales, save or load the files.

If you check the module you will find the code:

```
' define the sales structure
Public Structure SalesStruct
Dim Item As String
Dim Count As Integer
Dim Price As Single
Dim Total As Single
Dim TheDate As String
Dim Flg As Boolean
End Structure
```

Which defines the main structure, then

```
' define the main array to store sales info
Public Sales() As SalesStruct
Public SalesCount As Integer
```

Which defines the array and its number of elements

```
' read info
Public Function ReadSalesInfo() As SalesStruct
AddSale.TextBox1.Text = ""
AddSale.TextBox2.Text = ""
AddSale.TextBox3.Text = ""
AddSale.TextBox4.Text = ""
```

```
AddSale.TextBox5.Text = ""
If AddSale.ShowDialog = DialogResult.Cancel Then
    Dim dummy As SalesStruct
    dummy.Flg = False
    Return dummy
End If
Dim S As SalesStruct
S.Item = AddSale.TextBox1.Text
S.Total = AddSale.TextBox2.Text
S.TheDate = AddSale.TextBox4.Text
S.Flg = True
    Return S
End Function
```

This function uses a dialog called AddSale to read the information of an item. The first part just clears the text boxes on the form/dialog, and the if statement part shows the window and tells you if the user canceled the data entry, and the last part fills the structure from the form and returns the result.

```
' display the information of the strucutre in the data grid view
Public Sub DisplayArray(ByVal Arr() As SalesStruct, ByVal DGV As DataGridView)
DGV.Rows.Clear()
Dim I As Integer
For I = 0 To Arr.Length - 1
DGV.Rows.Add(Arr(I).Item, Arr(I).Count, Arr(I).Price, Arr(I).Total, Arr(I).TheDate)
Next
End Sub
```

This part displays the information of the array in a data grid view

```
' remove an item from array based on item name
   Public Sub RemoveItemBasedOnName(ByVal Name As String, ByRef Arr() As SalesStruct, ByRef IC As
Integer)
       Dim I As Integer
       Dim J As Integer
       For I = 0 To Arr.Length - 1
           If Name = Arr(I).Item Then
               For J = I + 1 To Arr.Length - 1
                   Arr(J - 1) = Arr(J)
               Next
                ReDim Preserve Arr(0 To Arr.Length - 2)
                IC = IC - 1
                Exit Sub
           End If
       Next
   End Sub
```

This one removes an item based on its name

```
' save the sales info
Public Sub SaveFile(ByVal FileName As String, ByVal Arr() As SalesStruct)
FileSystem.FileOpen(1, FileName, OpenMode.Output, OpenAccess.Write)
Dim I As Integer
FileSystem.PrintLine(1, Arr.Length)
For I = 0 To Arr.Length - 1
FileSystem.PrintLine(1, Arr(I).Item)
FileSystem.PrintLine(1, Arr(I).Price)
FileSystem.PrintLine(1, Arr(I).TheDate)
FileSystem.PrintLine(1, Arr(I).Total)
Next
FileSystem.FileClose(1)
End Sub
' load the file info
Public Sub LoadFile(ByVal FileName As String, ByRef Arr() As SalesStruct, ByRef IC As Integer)
```

```
FileSystem.FileOpen(1, FileName, OpenMode.Input, OpenAccess.Read)
Dim I As Integer
IC = FileSystem.LineInput(1)
ReDim Arr(0 To IC - 1)
For I = 0 To Arr.Length - 1
    Arr(I).Item = FileSystem.LineInput(1)
    Arr(I).Price = FileSystem.LineInput(1)
    Arr(I).TheDate = FileSystem.LineInput(1)
    Arr(I).Total = FileSystem.LineInput(1)
Next
FileSystem.FileClose(1)
End Sub
```

These two saves and load the information

```
' get total sum
Public Function GetTotalSales(ByVal Arr() As SalesStruct) As Single
  Dim S As Single = 0
  Dim I As Integer
  For I = 0 To Arr.Length - 1
      S += Arr(I).Total
      Next
      Return S
End Function
```

This last one finds the total. As you can see there is almost no difference in the code that is inside the module. It is exactly the same as the code you use in forms. Now if you open the main form of the application

| 🥠 TestSales - Microsoft Visual St | udio | | |
|-----------------------------------|---|---------------------------------------|----------------------------------|
| File Edit View Project E | uild Debug Data Format Tools Window Help | | |
| 1 🔂 • 🛅 • 🗃 🖬 🖉 1 🐰 1 | 🗅 📸 🖉 + 🔍 - 💭 + 🖏 🖡 🕨 Debug 🛛 + Any CPU 🚽 🎯 AutoLinkAccounts(by 🚽 🔩 😤 🐋 🖇 | č 💽 🖸 📲 | |
| ·(前) [[] 医 引] [] 必 與 | 日辺路寺事をあるためは国国のの国際。 | | |
| Toolbox - 4 X | | Solution Explorer - Solution 'TestSal | es' (1 project) - I X |
| All Windows Forms | Torini Louisidini Generalizi Start Page | BARBRAS | |
| 🖯 Common Controls | | Cabatian (TestCaber) (Testinat) | |
| Pointer | a Sales | TestSales (1 project) | |
| 💩 Button | File Sales Tools | My Project | |
| CheckBox | Item Count Price Total Date | - AddSale.vb | |
| 😫 CheckedListBox | | 🔄 Form1.vb | |
| ComboBox | | 🔚 General.vb | |
| DateTimePicker | | | |
| A Label | | | |
| A LinkLabel | | | |
| E ListBox | | | |
| 222 ListView | | | |
| #- MaskedTextBox | | | |
| MonthCalendar | | Solution Explorer Sclass View | |
| 5 NotifyIcon | | Properties | |
| 1 NumericUpDown | | Fioperaes | * # * |
| PictureBox | | Form1 System.Windows.Forms.For | m • |
| 🚥 ProgressBar | | | |
| RadioButton | J | Padding | 0, 0, 0, 0 |
| RichTextBox | | RightToLeft | No |
| abl TextBox | | RightToLeftLayout | False |
| 🗞 ToolTip | | Showlcon | True |
| TreeView | | ShowInTaskbar | True |
| 🛐 WebBrowser | | I Size | 507, 341 |
| Containers | | SizeGripStyle | Auto |
| 🗄 Menus & Toolbars | 🖆 OFD 🚊 MenuStrip1 📲 SFD | StartPosition | WindowsDefaultLocation |
| 🗄 Data | | Tag | |
| Components | | Text | Sales |
| Cutout | | 9 | |
| Ready | | | |
| | | | 1246 AM |
| S () | 💾 💾 🕔 🤍 🗶 🧮 🔍 🧐 🚳 🛍 👹 🌆 | | EN 🔺 🏴 🛱 🌓 12:46 AM 4/27/2010 |

This one contains a menu strip and a data grid view, with open files dialog and save file dialog. If you check the code of the form

```
Private Sub AddToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles AddToolStripMenuItem.Click
Dim SR As SalesStruct
SR = General.ReadSalesInfo
```

http://www.mka-soft.com

```
If SR.Flg Then
            SalesCount = SalesCount + 1
            ReDim Preserve Sales(0 To SalesCount - 1)
            Sales(SalesCount - 1) = SR
           DisplayArray(Sales, DGV)
       End If
   End Sub
   Private Sub RemoveSaleToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RemoveSaleToolStripMenuItem.Click
       If DGV.SelectedRows.Count = 0 Then
           Exit Sub
       End If
       RemoveItemBasedOnName(DGV.SelectedRows.Item(0).Cells(0).Value, Sales, SalesCount)
       DisplayArray(Sales, DGV)
   End Sub
   Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ExitToolStripMenuItem.Click
        End
   End Sub
   Private Sub SaToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles SaToolStripMenuItem.Click
       SFD.Filter = "*.txt |*.txt"
       If SFD.ShowDialog = Windows.Forms.DialogResult.Cancel Then
           Exit Sub
       End If
       SaveFile(SFD.FileName, Sales)
   End Sub
   Private Sub LoadToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles LoadToolStripMenuItem.Click
       If OFD.ShowDialog = Windows.Forms.DialogResult.Cancel Then
           Exit Sub
       End If
       LoadFile(OFD.FileName, Sales, SalesCount)
       DisplayArray(Sales, DGV)
   End Sub
   Private Sub FindTotalToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles FindTotalToolStripMenuItem.Click
       MsgBox("the total sales:" & GetTotalSales(Sales).ToString)
   End Sub
```

You can see the code is much smaller here, because it is just a call to the code in the module. In fact later on if you want to modify the user interface, the code of the module is not affected. Also since the code is much smaller, it is easier for other programmers to understand you code and update it.

So to sum things up, modules:

- 1- Are vb files
- 2- Used to store functions/subroutines, and other vb coding
- 3- Makes your program easier to maintin
- 4- Makes your program easier to understand
- 5- You can use/not use them, it is up to you
- 6- You can use any number of modules in a vb project
- 7- Make it easy to port your code to another application

8- Helps you isolate the interface design from program logic.

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: <u>notes@mka-soft.com</u>.

Thanks.

mkaatr