MenuInfo and Explorer Context Menus

Make an ExtenderProvider that adds images and different fonts to MenuItems in VB .NET

This example shows how to make an ExtenderProvider that adds images and different fonts to MenuItems in VB .NET. The provider catches the MeasureItem and DrawItem events for a MenuItem object and draws its image and text using the indicated font.

Append items to Windows Explorer Shell context menu easily – How ?

Add items to Windows Explorer context menu easily with Windows Explorer Shell Context Menu Gold. This powerful .Net component for custom items adding to Windows Explorer Shell context menu will add all your custom application entries to Explorer Shell context menu. This .Net component , C++ and Visual Basic .NET support include detailed C# / VB.NET samples, tutorials and support all you may need :
- Add items to Windows Explorer Shell context menu to be shown on any Windows computer (all OS are supported – Windows XP, Vista, x64 , etc.)
- Add all your items to Windows Explorer Shell context menu to be shown in any way - with your custom caption and icon, as separator or sub-menu
- Add your items to Windows Explorer Shell context menu to be shown for all types of files or shown only for files of particular type (for example, only for .DOC , .MP3,.WMA,.AAC , .AVI files)
- Add your program items to Windows Explorer Shell context menu, sub-menus, sub-menus of unlimited depth and even much more

Windows Explorer Shell Context Menu Gold - is a powerful .Net component that support all you may need to insert your program items to Windows Explorer Shell context menu - in a fast and a very easy way. Add your program entries to Windows Explorer Shell context menu right now – add entries to context menu fast and exactly as you prefere :

Introduction

MenuItems is an important element that was used in Windows OS of old versions to add custom entries to Windows Explorer Shell context menu, but because this method was used only for Windows 95 / Windows 98 (not on XP, Vista, x64 - 64-bit Windows), to add entries to Windows Explorer Shell context menu today you should use, according to Microsoft guidelines, appropriate .Net component - Windows Explorer Shell Context Menu Gold. This .Net component will add custom items of all types - separators, items with icon and without, items with custom text and command to Windows Explorer Shell context menu.

More about MenuItems

This provider stores MenuImage and MenuFont values for each of its client MenuItem objects in its private MenuInfo class. It stores MenuInfo objects in its m_MenuInfo hashtable. It also adds event handler to catch each client MenuItem's MeasureItem and DrawItem events.

When the MeasureItem event fires, the provider's event handler determines the space needed to display the MenuItem's image and text in the desired font.

When the DrawItem event fires, the provider's event handler draws the MenuItem's image and text. See the code for the details.

The GetMenuInfo helper function returns a MenuItem's MenuInfo object or a default object it the MenuItem doesn't have an entry.

The AddOrRemoveIfNecessary helper routine compares a MenuItem object to a default object. If the object has the default values, then the routine ensures that it is removed from the m_MenuInfo hashtable. If the object does not hold the default values, then the routine ensures that it is in the hashtable.

```
Imports System.ComponentModel

<ToolboxBitmap(GetType(MenuImageProvider), _
    "menu_image_provider.bmp"), _
 ProvideProperty("MenuImage", GetType(MenuItem)), _
 ProvideProperty("MenuFont", GetType(MenuItem))> _
Public Class MenuImageProvider
    Inherits System.ComponentModel.Component
    Implements IExtenderProvider

... Component Designer generated code ...

    ' Space between the image and text.
    Private Const IMAGE_SPACE As Integer = 5

    ' Information about a MenuItem.
    Private Class MenuInfo
        Public MenuImage As Image
        Public MenuFont As Font

        ' Return True if this object represents no range.
        Public Function IsDefault() As Boolean
            Return (MenuImage Is Nothing) And _
                   (MenuFont Is Nothing)
        End Function
    End Class

    ' The information about menus.
    Private m_MenuInfo As New Hashtable

    ' We can extend MenuItems.
    Public Function CanExtend(ByVal extendee As Object) As _
        Boolean Implements _
        System.ComponentModel.IExtenderProvider.CanExtend
        Return (TypeOf extendee Is MenuItem)
    End Function

    ' Return this MenuItem's MenuImage.
    <Category("Appearance"), _
     DefaultValue(GetType(Image), Nothing)> _
    Public Function GetMenuImage(ByVal menu_item As _
        MenuItem) As Image
        Return GetMenuInfo(menu_item).MenuImage
    End Function

    ' Set this control's minimum value.
    <Category("Appearance"), _
     DefaultValue(GetType(Image), Nothing)> _
    Public Sub SetMenuImage(ByVal menu_item As MenuItem, _
        ByVal menu_image As Image)
        ' Get the MenuItem's MenuInfo object.
        Dim menu_info As MenuInfo = GetMenuInfo(menu_item)

        ' See if the image is Nothing.
        If menu_image Is Nothing Then
            ' The image is Nothing.
            menu_info.MenuImage = menu_image
        Else
            ' The image is not Nothing.
```

```
        ' Copy it into a new Bitmap.
        Dim bm As New Bitmap(menu_image.Width, _
            menu_image.Height)
        Dim gr As Graphics = Graphics.FromImage(bm)
        gr.DrawImage(menu_image, 0, 0)

        ' Use the pixel in the upper left corner
        ' to set the image's transparent color.
        ' See if this pixel is already transparent.
        If bm.GetPixel(0, 0).A > 0 Then
            ' This pixel is not already transparent.
            ' Use it.
            bm.MakeTransparent(bm.GetPixel(0, 0))
        End If

        ' Set the new image.
        menu_info.MenuImage = bm
    End If

    ' Add or remove the MenuInfo if necessary.
    AddOrRemoveIfNecessary(menu_item, menu_info)
End Sub

' Return this MenuItem's MenuFont.
<Category("Appearance"), _
 DefaultValue(GetType(Font), Nothing)> _
Public Function GetMenuFont(ByVal menu_item As _
    MenuItem) As Font
    Return GetMenuInfo(menu_item).MenuFont
End Function

' Set this control's minimum value.
<Category("Appearance"), _
 DefaultValue(GetType(Font), Nothing)> _
Public Sub SetMenuFont(ByVal menu_item As MenuItem, _
    ByVal menu_font As Font)
    ' Get the MenuItem's MenuInfo object.
    Dim menu_info As MenuInfo = GetMenuInfo(menu_item)

    ' Set the new image.
    menu_info.MenuFont = menu_font

    ' Add or remove the MenuInfo if necessary.
    AddOrRemoveIfNecessary(menu_item, menu_info)
End Sub

' Return this MenuItem's MenuInfo.
Private Function GetMenuInfo(ByVal menu_item As _
    MenuItem) As MenuInfo
    ' See if we have MenuInfo for this control.
    If m_MenuInfo.Contains(menu_item) Then
        ' We have MenuInfo for this control. Return it.
        Return DirectCast(m_MenuInfo(menu_item), _
            MenuInfo)
    Else
        ' We do not have MenuInfo for this control.
        ' Return a new default MenuInfo.
        Return New MenuInfo
    End If
End Function

' Add or remove this MenuInfo if necessary.
Private Sub AddOrRemoveIfNecessary(ByVal menu_item As _
    MenuItem, ByVal menu_info As MenuInfo)
    ' See if the MenuInfo should be present but is not,
```

```
    ' or should not be present but is.
    If menu_info.IsDefault <> Not _
        m_MenuInfo.Contains(menu_item) Then
        If menu_info.IsDefault Then
            ' The MenuInfo should not be present but is.
            m_MenuInfo.Remove(menu_item)
            menu_item.OwnerDraw = False
            RemoveHandler menu_item.MeasureItem, _
                AddressOf Client_MeasureItem
            RemoveHandler menu_item.DrawItem, AddressOf _
                Client_DrawItem
        Else
            ' The MenuInfo should be present but is not.
            m_MenuInfo.Add(menu_item, menu_info)
            menu_item.OwnerDraw = True
            AddHandler menu_item.MeasureItem, AddressOf _
                Client_MeasureItem
            AddHandler menu_item.DrawItem, AddressOf _
                Client_DrawItem
        End If
    End If
End Sub

Private Sub Client_MeasureItem(ByVal sender As Object, _
    ByVal e As _
    System.Windows.Forms.MeasureItemEventArgs)
    Dim menu_item As MenuItem = DirectCast(sender, _
        MenuItem)
    Dim menu_info As MenuInfo = GetMenuInfo(menu_item)

    ' Get the size of the MenuImage.
    If Not (menu_info.MenuImage Is Nothing) Then
        e.ItemWidth = menu_info.MenuImage.Width + _
            IMAGE_SPACE
        e.ItemHeight = menu_info.MenuImage.Height
    End If

    ' Get the size of the text.
    Dim the_font As Font
    If (menu_info.MenuFont Is Nothing) Then
        the_font = menu_item.GetMainMenu.GetForm.Font
    Else
        the_font = menu_info.MenuFont
    End If
    Dim text_size As SizeF = _
        e.Graphics.MeasureString(menu_item.Text, _
        the_font)

    ' Add room for the text.
    e.ItemWidth += CInt(text_size.Width * 1.5)
    If e.ItemHeight < text_size.Height * 1.5 Then _
        e.ItemHeight = CInt(text_size.Height * 1.5)
End Sub

Private Sub Client_DrawItem(ByVal sender As Object, _
    ByVal e As System.Windows.Forms.DrawItemEventArgs)
    ' See if the item is selected.
    Dim fg_brush As Brush
    Dim bg_brush As Brush
    If (e.State And DrawItemState.Selected) = 0 Then
        ' Not selected.
        ' Use a light background and dark foreground.
        bg_brush = New SolidBrush(SystemColors.Menu)
        fg_brush = New SolidBrush(SystemColors.MenuText)
    Else
```

```
        ' Selected.
        ' Use a dark background and light foreground.
        bg_brush = New _
            SolidBrush(SystemColors.Highlight)
        fg_brush = New _
            SolidBrush(SystemColors.HighlightText)
    End If

    ' Draw the MenuItem's background.
    e.Graphics.FillRectangle(bg_brush, e.Bounds)

    ' Get the MenuItem and its MenuInfo data.
    Dim menu_item As MenuItem = DirectCast(sender, _
        MenuItem)
    Dim menu_info As MenuInfo = GetMenuInfo(menu_item)

    ' Draw the image.
    Dim x As Integer = e.Bounds.X
    Dim y As Integer = e.Bounds.Y
    If Not (menu_info.MenuImage Is Nothing) Then
        y += (e.Bounds.Height - _
            menu_info.MenuImage.Height) \ 2
        e.Graphics.DrawImage(menu_info.MenuImage, x, y)
        x += menu_info.MenuImage.Width + IMAGE_SPACE
    End If

    ' Draw the text.
    Dim the_font As Font
    If (menu_info.MenuFont Is Nothing) Then
        the_font = menu_item.GetMainMenu.GetForm.Font
    Else
        the_font = menu_info.MenuFont
    End If
    Dim layout_rect As New RectangleF( _
        x, e.Bounds.Y, _
        e.Bounds.Width - (x - e.Bounds.X), _
        e.Bounds.Height)
    Dim string_format As New StringFormat
    string_format.Alignment = StringAlignment.Near
    string_format.LineAlignment = StringAlignment.Center
    string_format.HotkeyPrefix = _
        System.Drawing.Text.HotkeyPrefix.Show
    e.Graphics.DrawString(menu_item.Text, the_font, _
        fg_brush, layout_rect, string_format)

    ' Free resources.
    fg_brush.Dispose()
    bg_brush.Dispose()
  End Sub
End Class
```

At design time, you set the IsRequiredMessage for TextBoxes that you want to be required. The main program doesn't need to use any code to validate TextBoxes.

If you want to ensure that all TextBoxes are filled in, call the HasError function in the form's Closing event handler to see if it is safe to close the form.

```
Private Sub Form1_Closing(ByVal sender As Object, ByVal e _
  As System.ComponentModel.CancelEventArgs) Handles _
  MyBase.Closing
  e.Cancel = IntegerRangeProvider1.HasError()
End Sub
```

```
Private Sub btnOk_Click(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles btnOk.Click
   Me.Close()
End Sub

Private Sub btnCancel_Click(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles btnCancel.Click
   IntegerRangeProvider1.Enabled = False
   Me.Close()
End Sub
```

The main program doesn't need any code to display the menus. Everything is taken care of by setting the MenuImage and MenuFont properties. The main program can handle the menus exactly as it handles other menus.

One drawback to this method is that the menus are not drawn at design time so you cannot see what they will look like until you run the program.