WinFBE – Visual Designer

The visual designer is still a major work in progress so please do not assume that all functionality is available and/or working correctly. The purpose of this release to show the progress of the visual designer to date and to allow input from users regarding its direction and whether changes in basic design philosophy should occur at this point.

NOTE: Code may fail, GPF's may occur in the editor. Do not rely on this version of the designer for any production quality code. Treat this mainly as a demo.

The following areas of the visual designer are not working:

- Menu Editor
- Toolbar Editor
- Statusbar Editor
- Lock Controls
- Images/Pictures/Icons. Any control property that relies on these types has not been implemented yet. Need to create an interface between the editor and resource file.
- Guidelines and snap to lines on the design form

There are only a few controls that have been implemented. More to come as the core structure of the designer itself matures.

- Forms
- Label
- Button (Jose Roca's CXpButton class)
- TextBox
- CheckBox
- ListBox (only parts of it so far)
- StatusBar (only through code at this time)

The forms and controls are being implemented via the WinFormsX library that I am writing and maintaining on GitHub.

The rest of this document will give you a bare bones introduction to the use of the visual designer and code to perform various tasks.

GENERAL OVERVIEW

The visual designer will automatically generate code whenever something changes related to a form or control. The generated code is based on the syntax of the WinFormsX library.

When the code is first generated, the visual designer will output an additional line:

```
Application.Run(Form1)
```

That line essentially bootstraps the form so that it will display. Obviously you don't need a line like this for every form that is generated. You can safely move that line to a main source file so that it only runs once at program startup. Any subsequent similar generated lines for other forms can be safely deleted.

WinFBE generates a TYPE structure for every form in the application. That TYPE extends a bass class called wfxForm which contains the core functionality of a form. You can find the wfxForm source code and all other source code for other controls such as wfxLabel, wfxTextBox in the WinFormsX include folder. A global shared variable is generated for each form TYPE. For example, if your form is named *Form1* then the shared variable is also named Form1.

```
Dim Shared Form1 as Form1Type
```

If you have controls on your form then you access them via dot syntax. For example, if you have a button called *Button1* located on the form called *Form1* then you access the button via *Form1.Button1*

Resource file – All WinFBE visual designer created applications need to have a resource file and manifest file. If you are using WinFBE's project management to create your program then the resource file and manifest file will be created automatically. If you are creating your application outside of a project then you need to manually provide a resource file that includes a reference to a manifest file. Failing to do this will result in your application not being theme aware. It will look ugly. If manually specifying a resource file then you can use WinFBE's built-in statement to identify the resource filename.

```
'#RESOURCE "resource.rc"
```

EVENT HANDLERS

```
Declare Function frmMain_Click( byref sender as wfxForm, byref e as EventArgs ) as
LRESULT
```

This is a declaration for an Event Handler. Event handlers are functions that are called in response to something occurring within the operating system or in response to an action by the user. For example, if the user clicks on a button or moves their mouse cursor over an area of the form or a control then an event is triggered. In Windows API language this is actually a message or notification generated by the operating system that then flows through the application's message pump eventually finding its way to the procedure handler for the application. If you have ever seen Win32 API style programs then you will recognize functions containing large SELECT CASE statements having such things as WM_COMMAND, WM_NOTIFY, WM_PAINT, WM_SIZE, etc. The visual designer hides all of the complexity of message pumps and that SELECT CASE from you by instead redirecting the message to an Event Handler that you define yourself.

Every Event Handler has the exact same format and is comprised of two parameters that must be defined as BYREF.

frmMain_Click: The portion before the underscore is the form name and the portion afterwards is the name of the event being responded to. In this case, the function is in response to someone clicking on an area of the form. The first parameter, *sender*, is a variable representing the form where the click occurred (eg. frmMain and is of the wfxForm class). You can interact with that variable and manipulate properties of the form:

```
sender.Text = "My new caption for the form's title bar!"
```

This is actually equivalent to using the form's explicitly defined shared variable and both are acceptable.

```
frmMain.Text = "My new caption for the form's title bar!"
```

When dealing with controls on a form, the syntax is only slightly different and pretty self-explanatory.

```
Declare Function frmMain_cmdOK_Click( byref sender as wfxButton, byref e as EventArgs
) as LRESULT
```

Of note is that *sender* is now of type *wfxButton* rather than *wfxForm*. This is because the function is in response to the user clicking on the OK button rather than the form itself.

The second parameter, *EventArgs*, is short for Event Arguments and is a variable that contains additional information related to the event being handled. For example, when handling mouse movement the EventArgs variable will contain the client coordinates of the mouse cursor and which (if any) mouse buttons are pressed. Likewise, responding to a KeyPress event the EventArgs variable would contain the character that was pressed and whether any Alt, Shift, or Ctrl keys are pressed. Get to know EventArgs because it is extremely convenient.

```
Type wfxEventArgs Extends Object
   private:
   public:
      Message As UINT        ' Windows value of message being sent (WM_COMMAND,
WM_NOTIFY, etc)
      wParam as WPARAM       ' the wParam of the raw message
      lParam as LPARAM       ' the lParam of the raw message
```

```
      Handled as Boolean     ' indicates whether the event is handled by the user
      Cancel as boolean      ' set to True to cancel closing of Form
      Ctrl as Boolean        ' the CTRL key is pressed
      Alt as Boolean         ' the Alt key is pressed
      Shift as Boolean       ' the SHIFT key is pressed
      KeyChar as long        ' stores the character corresponding to the key pressed
      KeyCode as long        ' stores the keyboard code for the event
      LButton as boolean     ' the left mouse button pressed
      MButton as boolean     ' the middle mouse button pressed
      RButton as boolean     ' the right mouse button pressed
      x as long              ' the x-coordinate of the mouse click
      y as long              ' the y-coordinate of the mouse click
      PanelClickIndex as long   ' the zero-based index of clicked statusbar panel
      hDrop as HDROP         ' handle used for WM_DROPFILES message
end type
```

Colors

```
TYPE wfxColors
   SystemActiveBorder      As COLORREF = GetSysColor(COLOR_ACTIVEBORDER)
   SystemActiveCaption     As COLORREF = GetSysColor(COLOR_ACTIVECAPTION)
   SystemActiveCaptionText As COLORREF = GetSysColor(COLOR_CAPTIONTEXT)
   SystemAppWorkspace      As COLORREF = GetSysColor(COLOR_APPWORKSPACE)
   SystemButtonFace        As COLORREF = GetSysColor(COLOR_BTNFACE)
   SystemButtonHighlight   As COLORREF = GetSysColor(COLOR_BTNHILIGHT)
   SystemButtonShadow      As COLORREF = GetSysColor(COLOR_BTNSHADOW)
   SystemControl           As COLORREF = GetSysColor(COLOR_3DFACE)
   SystemControlDark       As COLORREF = GetSysColor(COLOR_3DSHADOW)
   SystemControlDarkDark   As COLORREF = GetSysColor(COLOR_3DDKSHADOW)
   SystemControlLight      As COLORREF = GetSysColor(COLOR_3DLIGHT)
   SystemControlLightLight As COLORREF = GetSysColor(COLOR_3DHILIGHT)
   SystemControlText       As COLORREF = GetSysColor(COLOR_BTNTEXT)
   SystemDesktop           As COLORREF = GetSysColor(COLOR_DESKTOP)
   SystemGradientActiveCaption   As COLORREF =
GetSysColor(COLOR_GRADIENTACTIVECAPTION)
   SystemGradientInactiveCaption as COLORREF =
GetSysColor(COLOR_GRADIENTINACTIVECAPTION)
   SystemGrayText          As COLORREF = GetSysColor(COLOR_GRAYTEXT)
   SystemHighlight         As COLORREF = GetSysColor(COLOR_HIGHLIGHT)
   SystemHighlightText     As COLORREF = GetSysColor(COLOR_HIGHLIGHTTEXT)
   SystemHotTrack          As COLORREF = GetSysColor(COLOR_HOTLIGHT)
   SystemInactiveBorder    As COLORREF = GetSysColor(COLOR_INACTIVEBORDER)
   SystemInactiveCaption   As COLORREF = GetSysColor(COLOR_INACTIVECAPTION)
   SystemInactiveCaptionText As COLORREF = GetSysColor(COLOR_INACTIVECAPTIONTEXT)
   SystemInfo              As COLORREF = GetSysColor(COLOR_INFOBK)
   SystemInfoText          As COLORREF = GetSysColor(COLOR_INFOTEXT)
   SystemMenu              As COLORREF = GetSysColor(COLOR_MENU)
   SystemMenuBar           As COLORREF = GetSysColor(COLOR_MENUBAR)
   SystemMenuHighlight     As COLORREF = GetSysColor(COLOR_MENUHILIGHT)
   SytemMenuText           As COLORREF = GetSysColor(COLOR_MENUTEXT)
   SystemScrollBar         As COLORREF = GetSysColor(COLOR_SCROLLBAR)
   SystemWindow            As COLORREF = GetSysColor(COLOR_WINDOW)
   SystemWindowFrame       As COLORREF = GetSysColor(COLOR_WINDOWFRAME)
   SystemWindowText        As COLORREF = GetSysColor(COLOR_WINDOWTEXT)

   AliceBlue            As COLORREF = BGR(240,248,255)
   AntiqueWhite         as COLORREF = BGR(250,235,215)
   Aqua                 as COLORREF = BGR(  0,255,255)
   Aquamarine           as COLORREF = BGR(127,255,212)
   Azure                as COLORREF = BGR(240,255,255)
   Beige                as COLORREF = BGR(245,245,220)
   Bisque               as COLORREF = BGR(255,228,196)
   Black                as COLORREF = BGR(  0,  0,  0)
   BlanchedAlmond       as COLORREF = BGR(255,255,205)
   Blue                 as COLORREF = BGR(  0,  0,255)
   BlueViolet           as COLORREF = BGR(138, 43,226)
   Brown                as COLORREF = BGR(165, 42, 42)
   Burlywood            as COLORREF = BGR(222,184,135)
   CadetBlue            as COLORREF = BGR( 95,158,160)
   Chartreuse           as COLORREF = BGR(127,255,  0)
   Chocolate            as COLORREF = BGR(210,105, 30)
   Coral                as COLORREF = BGR(255,127, 80)
   CornflowerBlue       as COLORREF = BGR(100,149,237)
   Cornsilk             as COLORREF = BGR(255,248,220)
```

```
Crimson              as COLORREF = BGR(220, 20, 60)
Cyan                 as COLORREF = BGR(  0,255,255)
DarkBlue             as COLORREF = BGR(  0,  0,139)
DarkCyan             as COLORREF = BGR(  0,139,139)
DarkGoldenRod        as COLORREF = BGR(184,134, 11)
DarkGray             as COLORREF = BGR(169,169,169)
DarkGreen            as COLORREF = BGR(  0,100,  0)
DarkKhaki            as COLORREF = BGR(189,183,107)
DarkMagenta          as COLORREF = BGR(139,  0,139)
DarkOliveGreen       as COLORREF = BGR( 85,107, 47)
DarkOrange           as COLORREF = BGR(255,140,  0)
DarkOrchid           as COLORREF = BGR(153, 50,204)
DarkRed              as COLORREF = BGR(139,  0,  0)
DarkSalmon           as COLORREF = BGR(233,150,122)
DarkSeaGreen         as COLORREF = BGR(143,188,143)
DarkSlateBlue        as COLORREF = BGR( 72, 61,139)
DarkSlateGray        as COLORREF = BGR( 47, 79, 79)
DarkTurquoise        as COLORREF = BGR(  0,206,209)
DarkViolet           as COLORREF = BGR(148,  0,211)
DeepPink             as COLORREF = BGR(255, 20,147)
DeepSkyBlue          as COLORREF = BGR(  0,191,255)
DimGray              as COLORREF = BGR(105,105,105)
DodgerBlue           as COLORREF = BGR( 30,144,255)
FireBrick            as COLORREF = BGR(178, 34, 34)
FloralWhite          as COLORREF = BGR(255,250,240)
ForestGreen          as COLORREF = BGR( 34,139, 34)
Fuchsia              as COLORREF = BGR(255,  0,255)
Gainsboro            as COLORREF = BGR(220,220,220)
GhostWhite           as COLORREF = BGR(248,248,255)
Gold                 as COLORREF = BGR(255,215,  0)
GoldenRod            as COLORREF = BGR(218,165, 32)
Gray                 as COLORREF = BGR(127,127,127)
Green                as COLORREF = BGR(  0,128,  0)
GreenYellow          as COLORREF = BGR(173,255, 47)
HoneyDew             as COLORREF = BGR(240,255,240)
HotPink              as COLORREF = BGR(255,105,180)
IndianRed            as COLORREF = BGR(205, 92, 92)
Indigo               as COLORREF = BGR( 75,  0,130)
Ivory                as COLORREF = BGR(255,255,240)
Khaki                as COLORREF = BGR(240,230,140)
Lavender             as COLORREF = BGR(230,230,250)
LavenderBlush        as COLORREF = BGR(255,240,245)
Lawngreen            as COLORREF = BGR(124,252,  0)
LemonChiffon         as COLORREF = BGR(255,250,205)
LightBlue            as COLORREF = BGR(173,216,230)
LightCoral           as COLORREF = BGR(240,128,128)
LightCyan            as COLORREF = BGR(224,255,255)
LightGoldenRodYellow as COLORREF = BGR(250,250,210)
LightGreen           as COLORREF = BGR(144,238,144)
LightGrey            as COLORREF = BGR(211,211,211)
LightPink            as COLORREF = BGR(255,182,193)
LightSalmon          as COLORREF = BGR(255,160,122)
LightSeaGreen        as COLORREF = BGR( 32,178,170)
LightSkyBlue         as COLORREF = BGR(135,206,250)
LightSlateGray       as COLORREF = BGR(119,136,153)
LightSteelBlue       as COLORREF = BGR(176,196,222)
LightYellow          as COLORREF = BGR(255,255,224)
Lime                 as COLORREF = BGR(  0,255,  0)
```

```
LimeGreen            as COLORREF = BGR( 50,205, 50)
Linen                as COLORREF = BGR(250,240,230)
Magenta              as COLORREF = BGR(255,  0,255)
Maroon               as COLORREF = BGR(128,  0,  0)
MediumAquamarine     as COLORREF = BGR(102,205,170)
MediumBlue           as COLORREF = BGR(  0,  0,205)
MediumOrchid         as COLORREF = BGR(186, 85,211)
MediumPurple         as COLORREF = BGR(147,112,219)
MediumSeaGreen       as COLORREF = BGR( 60,179,113)
MediumSlateBlue      as COLORREF = BGR(123,104,238)
MediumSpringGreen    as COLORREF = BGR(  0,250,154)
MediumTurquoise      as COLORREF = BGR( 72,209,204)
MediumVioletRed      as COLORREF = BGR(199, 21,133)
MidnightBlue         as COLORREF = BGR( 25, 25,112)
MintCream            as COLORREF = BGR(245,255,250)
MistyRose            as COLORREF = BGR(255,228,225)
Moccasin             as COLORREF = BGR(255,228,181)
NavajoWhite          as COLORREF = BGR(255,222,173)
Navy                 as COLORREF = BGR(  0,  0,128)
Navyblue             as COLORREF = BGR(159,175,223)
OldLace              as COLORREF = BGR(253,245,230)
Olive                as COLORREF = BGR(128,128,  0)
OliveDrab            as COLORREF = BGR(107,142, 35)
Orange               as COLORREF = BGR(255,165,  0)
OrangeRed            as COLORREF = BGR(255, 69,  0)
Orchid               as COLORREF = BGR(218,112,214)
PaleGoldenRod        as COLORREF = BGR(238,232,170)
PaleGreen            as COLORREF = BGR(152,251,152)
PaleTurquoise        as COLORREF = BGR(175,238,238)
PaleVioletRed        as COLORREF = BGR(219,112,147)
PapayaWhip           as COLORREF = BGR(255,239,213)
PeachPuff            as COLORREF = BGR(255,218,185)
Peru                 as COLORREF = BGR(205,133, 63)
Pink                 as COLORREF = BGR(255,192,203)
Plum                 as COLORREF = BGR(221,160,221)
PowderBlue           as COLORREF = BGR(176,224,230)
Purple               as COLORREF = BGR(128,  0,128)
Red                  as COLORREF = BGR(255,  0,  0)
RosyBrown            as COLORREF = BGR(188,143,143)
RoyalBlue            as COLORREF = BGR( 65,105,225)
SaddleBrown          as COLORREF = BGR(139, 69, 19)
Salmon               as COLORREF = BGR(250,128,114)
SandyBrown           as COLORREF = BGR(244,164, 96)
SeaGreen             as COLORREF = BGR( 46,139, 87)
SeaShell             as COLORREF = BGR(255,245,238)
Sienna               as COLORREF = BGR(160, 82, 45)
Silver               as COLORREF = BGR(192,192,192)
SkyBlue              as COLORREF = BGR(135,206,235)
SlateBlue            as COLORREF = BGR(106, 90,205)
SlateGray            as COLORREF = BGR(112,128,144)
Snow                 as COLORREF = BGR(255,250,250)
SpringGreen          as COLORREF = BGR(  0,255,127)
SteelBlue            as COLORREF = BGR( 70,130,180)
Tan                  as COLORREF = BGR(210,180,140)
Teal                 as COLORREF = BGR(  0,128,128)
Thistle              as COLORREF = BGR(216,191,216)
Tomato               as COLORREF = BGR(255, 99, 71)
Turquoise            as COLORREF = BGR( 64,224,208)
```

```
   Violet               as COLORREF = BGR(238,130,238)
   Wheat                as COLORREF = BGR(245,222,179)
   White                as COLORREF = BGR(255,255,255)
   WhiteSmoke           as COLORREF = BGR(245,245,245)
   Yellow               as COLORREF = BGR(255,255,  0)
   YellowGreen          as COLORREF = BGR(139,205, 50)
end Type

''
''  Create system wide access to this colors class
Dim Shared Colors as wfxColors
```

FONTS

Creating and assigning a new font

```
this.CheckBox1.Font = New wfxFont("Segoe UI",9,FontStyles.Bold,FontCharset.Ansi)
```

Parameters

FontName

FontSize

FontStyle (use Or to combine multiple styles)

```
FontStyles.Normal
FontStyles.Bold
FontStyles.Italic
FontStyles.Strikeout
FontStyles.Underline
```

CharacterSet

```
FontCharset.Default
FontCharset.Ansi
FontCharset.Arabic
FontCharset.Baltic
FontCharset.ChineseBig5
FontCharset.EastEurope
FontCharset.GB2312
FontCharset.Greek
FontCharset.Hangul
FontCharset.Hebrew
FontCharset.Johab
FontCharset.Mac
FontCharset.OEM
FontCharset.Russian
FontCharset.Shiftjis
FontCharset.Symbol
FontCharset.Thai
FontCharset.Turkish
FontCharset.Vietnamese
```

ENUMS

```
FormWindowState.Maximized
FormWindowState.Minimized
FormWindowState.Normal

FormStartPosition.CenterParent
FormStartPosition.CenterScreen
FormStartPosition.Manual
FormStartPosition.WindowsDefaultLocation

FormBorderStyle.None
FormBorderStyle.Sizable
FormBorderStyle.Fixed3D
FormBorderStyle.FixedSingle
FormBorderStyle.FixedDialog
FormBorderStyle.FixedToolWindow
FormBorderStyle.SizableToolWindow

ControlBorderStyle.None
ControlBorderStyle.FixedSingle
ControlBorderStyle.Fixed3D

ImageLayout.None
ImageLayout.Tile
ImageLayout.Center
ImageLayout.Stretch
ImageLayout.Zoom

ButtonAlignment.BottomCenter
ButtonAlignment.BottomLeft
ButtonAlignment.BottomRight
ButtonAlignment.MiddleCenter
ButtonAlignment.MiddleLeft
ButtonAlignment.MiddleRight
ButtonAlignment.TopCenter
ButtonAlignment.TopLeft
ButtonAlignment.TopRight

LabelAlignment.MiddleCenter
LabelAlignment.MiddleLeft
LabelAlignment.MiddleRight
LabelAlignment.TopCenter
LabelAlignment.TopLeft
LabelAlignment.TopRight

CharacterCase.Normal
CharacterCase.Upper
CharacterCase.Lower

ScrollBars.None
ScrollBars.Horizontal
ScrollBars.Vertical
ScrollBars.Both

CheckBoxState.Checked
CheckBoxState.Unchecked
CheckBoxState.Indeterminate
```

```
TextAlignment.Left
TextAlignment.Right
TextAlignment.Center

ListSelectionMode.None
ListSelectionMode.One
ListSelectionMode.MultiSimple
ListSelectionMode.MultiExtended
```

FORMS

Properties

AcceptButton - Gets or sets a reference to the button that receives a click message when Enter key is pressed.
AllowDrop - Gets or sets a value (true/false) indicating whether the control will accept data that is dragged onto it.
BackColor - Gets or sets the background color of the form. Refer to the Colors object.
BackgroundImage – Gets or set the image to display in the background.
BackgroundImageLayout – Gets or sets the layout position of the background image. Refer to the ImageLayout enum.
BorderStyle - Gets or sets the border style of the form. Refer to the FormBorderStyle enum.
CancelButton - Gets or sets a reference to the button that receives a click message when Escape key is pressed.
ClientSize - Gets or sets the client area of the form. The client area of the form is the size of the form excluding the borders and the title bar. Get: returns wfxSize object. Set: (width, height)
ControlBox - Gets or sets value (true/false) indicating whether a control box is displayed in the caption bar of the form.
CtrlType - Gets or sets the control type value. Always ControlType.Form and used when adding form to the application's form collection.
Enabled - Gets or sets a value (true/false) indicating whether the form can respond to user interaction.
Height - Gets or sets the height of the form.
hWindow - Gets the Windows handle (hwnd) of the form.
Icon – Gets or sets the icon to display in the form's system menu box.
IsMainForm - Gets or sets a value (true/false) indicating the form is main and will display when application starts. When the form is closed the application also ends.
IsModal - Gets a value (true/false) indicating whether the form is displayed modally.
Left - Gets or sets the distance, in pixels, between the left edge of the form and the left edge of its container's client area (normally the screen).
Location - Gets or sets the top and left position of the form. Get: returns wfxPoint object. Set: (left, top).
Locked - Gets or sets a value (true/false) indicating whether the control can be moved or resized.
MaximizeBox - Gets or sets a value (true/false) indicating whether the maximize button is displayed in the caption bar of the form.
MaximumHeight – Gets or sets the maximum height of the form.
MaximumWidth – Gets or sets the maximum width of the form.
MinimizeBox - Gets or sets a value (true/false) indicating whether the minimize button is displayed in the caption bar of the form.
MinimumHeight - Gets or sets the minimum height of the form.
MinimumWidth- Gets or sets the minimum width of the form.
Parent - Gets or sets the parent container of the form.
ShowInTaskbar – Gets or sets a vaue (true/false) indicating whether the form appears in the Windows Taskbar.
Size - Gets or sets the size of the form. Get: returns wfxSize object. Set: (width, height)
StartPosition - Gets or sets the starting position of the form at run time. Refer to FormStartPosition

enum.
Tag – Gets or sets user defined text associated with the form.
Text - Gets or sets the text (caption) associated with this form.
Top - Gets or sets the distance, in pixels, between the top edge of the form and the top edge of its container's client area (normally the screen).
Visible - Gets or sets a value (true/false) indicating whether the form is displayed.
Width - Gets or sets the width of the form.
WindowState - Gets or sets a value that indicates whether form is minimized, maximized, or normal. Refer to the FormWindowState enum.

Methods

Close - Closes the form.
Hide - Conceals the form from the user.
Refresh - Forces the form to invalidate its client area and immediately redraw itself and any child controls.
SetBounds - Sets the bounds of the form to the specified location and size. (left, top, width, height).
Show – Creates and displays the form to the user.
ShowDialog – Creates and shows the form as a modal dialog box. |

Events

OnActivated - Occurs when the form is activated in code or by the user.
OnAllEvents - Special handler where all events are routed through. Use this handler if you prefer to use the Win32 api style messages and wParam and lParam parameters. Set the *Handled* element of *EventArgs* to *true* if you handle a message and do not want Windows to perform any further processing on the message.
OnClick - Occurs when the client area of the form is clicked.
OnDeactivate - Occurs when the form loses focus and is no longer the active form.
OnFormClosed - Occurs after the form is closed.
OnFormClosing - Occurs before the form is closed.
OnKeyDown - Occurs when a key is pressed while the form has focus.
OnKeyPress - Occurs when a character, space or backspace key is pressed while the form has focus.
OnKeyUp - Occurs when a key is released while the form has focus.
OnLoad - Occurs before a form is displayed for the first time.
OnMouseDoubleClick - Occurs when the form is double clicked by the mouse.
OnMouseDown - Occurs when the mouse pointer is over the form and a mouse button is pressed.
OnMouseEnter - Occurs when the mouse pointer enters the form.
OnMouseHover - Occurs when the mouse pointer rests on the form.
OnMouseLeave - Occurs when the mouse pointer leaves the form.
OnMouseMove - Occurs when the mouse pointer is moved over the form.
OnMouseUp - Occurs when the mouse pointer is over the form and a mouse button is released.
OnMove - Occurs when the form is moved.

OnResize - Occurs when the form is resized.
OnShown - Occurs whenever the form is first displayed.

Form events in a specific order every time a form is created and shown.

During form creation:

OnFormLoad: The form handle and all child controls exist however the form and controls are not yet visible. Respond to this event to reposition controls or to add data to controls. For example, add rows to a Listbox or Combobox.

OnFormActivated: The form has gained input focus (similar to the OnGotFocus event of a control).

OnShown: This event is only raised the first time a form is displayed; subsequently minimizing, maximizing, restoring, hiding, showing, or invalidating and repainting will not raise this event.

During form destruction:

OnFormClosing: Event occurs as the form is closing. If you cancel this event, then the form remains open. To cancel, simply set the Cancel element of the EventArgs structure to True.

OnDeactivate: Occurs when the form loses focus and is no longer the active form (similar to the OnLostFocus event of a control).

OnFormClosed: Occurs after the form has closed (similar to the OnDestroy event of a control).

How to…

Display a popup modal form

Use the *ShowDialog* method. Pass the form that is the parent of the popup (in the example below that would be frmMain).

```
frmPopup.ShowDialog( frmMain )
```

LABELS

Properties

AllowDrop - Gets or sets a value (true/false) indicating whether the control will accept data that is dragged onto it.
BackColor - Gets or sets the background color of the control. Refer to the Colors object.
BorderStyle - Gets or sets the border style of the label. Refer to the ControlBorderStyle enum.
CtrlID – Gets or sets a value indicating the control ID of the control.
CtrlType - Gets or sets the control type value. Always ControlType.Label and used when adding control to its form's controls collection.
Enabled - Gets or sets a value (true/false) indicating whether the control can respond to user interaction.
Font - Gets or sets the font for the control. Refer to the Font object.
ForeColor - Gets or sets the background color of the control. Refer to the Colors object.
Height - Gets or sets the height of the control.
hWindow - Gets the Windows handle (hwnd) of the control.
Left - Gets or sets the distance, in pixels, between the left edge of the control and the left edge of its container's client area (normally the form).
Location - Gets or sets the top and left position of the form. Get: returns wfxPoint object. Set: (left, top).
Locked - Gets or sets a value (true/false) indicating whether the control can be moved or resized.
Parent - Gets or sets the parent container of the control.
Size - Gets or sets the size of the control. Get: returns wfxSize object. Set: (width, height).
Tag – Gets or sets user defined text associated with the control.
Text - Gets or sets the text (caption) associated with this control.
TextAlign - Gets or sets a value indicating the alignment of the text on a control. Refer to LabelAlignment enum.
Top - Gets or sets the distance, in pixels, between the top edge of the control and the top edge of its container's client area (normally the form).
UseMnemonic – Gets or sets a value (true/false) indicating whether the first character preceded by an ampersand (&) character will be used as the control's accelerator key.
Visible - Gets or sets a value (true/false) indicating whether the control is displayed.
Width - Gets or sets the width of the control.

Methods

Hide - Conceals the control from the user.
Refresh - Forces the control to invalidate its client area and immediately redraw itself.
SelectNextControl – Moves the input control to the next (True) or previous (False) control in the tab order.
SetBounds - Sets the bounds of the control to the specified location and size. (left, top, width, height).
Show – Creates and makes the control visible to the user.

Events

OnAllEvents - Special handler where all events are routed through. Use this handler if you prefer to use the Win32 api style messages and wParam and lParam parameters. Set the Handled element of EventArgs to True if you handle a message and do not want Windows to perform any further processing on the message.
OnClick - Occurs when the client area of the control is clicked.
OnDestroy - Occurs immediately before the control is about to be destroyed and all resources associated with it released.
OnMouseDoubleClick - Occurs when the control is double clicked by the mouse.
OnMouseDown - Occurs when the mouse pointer is over the control and a mouse button is pressed.
OnMouseEnter - Occurs when the mouse pointer enters the control.
OnMouseHover - Occurs when the mouse pointer rests on the control.
OnMouseLeave - Occurs when the mouse pointer leaves the control.
OnMouseMove - Occurs when the mouse pointer is moved over the control.
OnMouseUp - Occurs when the mouse pointer is over the control and a mouse button is released.

BUTTON

Properties

AllowDrop - Gets or sets a value (true/false) indicating whether the control will accept data that is dragged onto it.
BackColor - Gets or sets the background color of the control. Refer to the Colors object.
CtrlID – Gets or sets a value indicating the control ID of the control.
CtrlType - Gets or sets the control type value. Always ControlType.Button and used when adding control to its form's controls collection.
Enabled - Gets or sets a value (true/false) indicating whether the control can respond to user interaction.
Focused - Gets or sets a value (true/false) indicating whether the control has input focus.
Font - Gets or sets the font for the control. Refer to the Font object.
ForeColor - Gets or sets the background color of the control. Refer to the Colors object.
Height - Gets or sets the height of the control.
hWindow - Gets the Windows handle (hwnd) of the control.
Image – Gets or set the image to display.
ImageLayout – Gets or sets the layout position of the image. Refer to the ImageLayout enum.
Left - Gets or sets the distance, in pixels, between the left edge of the control and the left edge of its container's client area (normally the form).
Location - Gets or sets the top and left position of the form. Get: returns wfxPoint object. Set: (left, top).
Locked - Gets or sets a value (true/false) indicating whether the control can be moved or resized.
Parent - Gets or sets the parent container of the control.
Size - Gets or sets the size of the control. Get: returns wfxSize object. Set: (width, height).
TabIndex - Gets or sets the position that the control occupies in the TAB position.
TabStop - Gets or sets a value (true/false) indicating whether the user can use the TAB key to give focus to the control.
Tag – Gets or sets user defined text associated with the control.
Text - Gets or sets the text (caption) associated with this control.
TextAlign - Gets or sets a value indicating the alignment of the text on a control. Refer to ButtonAlignment enum.
TextBackColor – Gets or sets the background color of the control. ThemeSupport property must be set to False.
TextBackColorDown - Gets or sets the background color of the control when the button is pressed. ThemeSupport property must be set to False.
TextForeColor - Gets or sets the foreground color of the control. ThemeSupport property must be set to False.
TextForeColorDown - Gets or sets the foreground color of the control when the button is pressed. ThemeSupport property must be set to False.
TextMargin – Gets or sets a value indicating the margin in pixels to the text of a button control.
ThemeSupport - Gets or sets a value (true/false) indicating whether Windows Theme will be applied to the control.
ToggleMode - Gets or sets a value (true/false) indicating whether the button allows dual states to toggle between On and Off.

Top - Gets or sets the distance, in pixels, between the top edge of the control and the top edge of its container's client area (normally the form).
UseMnemonic – Gets or sets a value (true/false) indicating whether the first character preceded by an ampersand (&) character will be used as the control's accelerator key.
Visible - Gets or sets a value (true/false) indicating whether the control is displayed.
Width - Gets or sets the width of the control.

Methods

Hide - Conceals the control from the user.
Refresh - Forces the control to invalidate its client area and immediately redraw itself.
SelectNextControl – Moves the input control to the next (True) or previous (False) control in the tab order.
SetBounds - Sets the bounds of the control to the specified location and size. (left, top, width, height).
Show – Creates and makes the control visible to the user.

Events

OnAllEvents - Special handler where all events are routed through. Use this handler if you prefer to use the Win32 api style messages and wParam and lParam parameters. Set the Handled element of EventArgs to True if you handle a message and do not want Windows to perform any further processing on the message.
OnClick - Occurs when the client area of the control is clicked.
OnDestroy - Occurs immediately before the control is about to be destroyed and all resources associated with it released.
OnMouseDoubleClick - Occurs when the control is double clicked by the mouse.
OnMouseDown - Occurs when the mouse pointer is over the control and a mouse button is pressed.
OnMouseEnter - Occurs when the mouse pointer enters the control.
OnMouseHover - Occurs when the mouse pointer rests on the control.
OnMouseLeave - Occurs when the mouse pointer leaves the control.
OnMouseMove - Occurs when the mouse pointer is moved over the control.
OnMouseUp - Occurs when the mouse pointer is over the control and a mouse button is released.

TEXTBOX

Properties

AcceptsReturn - Gets or sets a value (true/false) indicating whether the multiline textbox will accept the Enter character as input.
AcceptsTab - Gets or sets a value (true/false) indicating whether the multiline textbox will accept the Tab character as input.
AllowDrop - Gets or sets a value (true/false) indicating whether the control will accept data that is dragged onto it.
BackColor - Gets or sets the background color of the control. Refer to the Colors object.
BorderStyle - Gets or sets the border style of the control. Refer to the ControlBorderStyle enum.
CharacterCasing - Gets or sets a value indicating the case of the text. Refer to the CharacterCasing enum.
CtrlID – Gets or sets a value indicating the control ID of the control.
CtrlType - Gets or sets the control type value. Always ControlType.TextBox and used when adding control to its form's controls collection.
Enabled - Gets or sets a value (true/false) indicating whether the control can respond to user interaction.
Focused - Gets or sets a value (true/false) indicating whether the control has input focus.
Font - Gets or sets the font for the control. Refer to the Font object.
ForeColor - Gets or sets the background color of the control. Refer to the Colors object.
Height - Gets or sets the height of the control.
HideSelection - Gets or sets a value (true/false) indicating whether the selection should be hidden when the control loses focus.
hWindow - Gets the Windows handle (hwnd) of the control.
Left - Gets or sets the distance, in pixels, between the left edge of the control and the left edge of its container's client area (normally the form).
Location - Gets or sets the top and left position of the form. Get: returns wfxPoint object. Set: (left, top).
Locked - Gets or sets a value (true/false) indicating whether the control can be moved or resized.
MaxLength - Gets or sets a value indicating the maximum number of characters that can be entered into the control.
MultiLine - Gets or sets a value (true/false) indicating whether the text in the edit control can span more than one line.
Parent - Gets or sets the parent container of the control.
PasswordChar - Gets or sets a character to display for password input in a single line edit control.
ReadOnly - Gets or sets a value (true/false) indicating whether the text can be edited.
SelectionStart - Gets or sets a value indicating the start of the selected text.
SelectionLength - Gets or sets a value indicating the length of the selected text.
Size - Gets or sets the size of the control. Get: returns wfxSize object. Set: (width, height).
TabIndex - Gets or sets the position that the control occupies in the TAB position.
TabStop - Gets or sets a value (true/false) indicating whether the user can use the TAB key to give focus to the control.
Tag – Gets or sets user defined text associated with the control.
Text - Gets or sets the text (caption) associated with this control.

TextAlign - Gets or sets a value indicating the alignment of the text on a control. Refer to TextAlignment enum.
TextScrollBars – Gets or sets the value to indicate what scrollbars to display. Refer to the ScrollBars enum.
Top - Gets or sets the distance, in pixels, between the top edge of the control and the top edge of its container's client area (normally the form).
Visible - Gets or sets a value (true/false) indicating whether the control is displayed.
Width - Gets or sets the width of the control.
WordWrap - Gets or sets a value (true/false) indicating whether text is automatically word wrapped for multiline controls.

Methods

Hide - Conceals the control from the user.
Refresh - Forces the control to invalidate its client area and immediately redraw itself.
SelectNextControl – Moves the input control to the next (True) or previous (False) control in the tab order.
SetBounds - Sets the bounds of the control to the specified location and size. (left, top, width, height).
Show – Creates and makes the control visible to the user.

Events

OnAllEvents - Special handler where all events are routed through. Use this handler if you prefer to use the Win32 api style messages and wParam and lParam parameters. Set the Handled element of EventArgs to True if you handle a message and do not want Windows to perform any further processing on the message.
OnClick - Occurs when the client area of the control is clicked.
OnDestroy - Occurs immediately before the control is about to be destroyed and all resources associated with it released.
OnMouseDoubleClick - Occurs when the control is double clicked by the mouse.
OnMouseDown - Occurs when the mouse pointer is over the control and a mouse button is pressed.
OnMouseEnter - Occurs when the mouse pointer enters the control.
OnMouseHover - Occurs when the mouse pointer rests on the control.
OnMouseLeave - Occurs when the mouse pointer leaves the control.
OnMouseMove - Occurs when the mouse pointer is moved over the control.
OnMouseUp - Occurs when the mouse pointer is over the control and a mouse button is released.

How to…

Allow the Enter key to simulate pressing the Tab key

```
''
''
Function frmMain_txtAddress_KeyUp( ByRef sender As wfxTextBox, ByRef e As EventArgs)
As LRESULT
```

```
   ' Catch the RETURN key to simulate TAB and Shift-TAB
   ' If you were wanting to test multiple textboxes controls (eg. a series of
textboxes) then
   ' I would the following code in a dedicated function rather than duplicating it
each time.
   if e.KeyCode = VK_RETURN then
      if e.Shift then
         sender.SelectNextControl(false)
      else
         sender.SelectNextControl(true)
      end if
      e.Handled = true
   END IF

   Function = 0
End Function
```

CHECKBOX

Properties

AllowDrop - Gets or sets a value (true/false) indicating whether the control will accept data that is dragged onto it.
BackColor - Gets or sets the background color of the control. Refer to the Colors object.
CheckState - Gets or sets the state of the control. Refer to the CheckBoxState enum.
CtrlID – Gets or sets a value indicating the control ID of the control.
CtrlType - Gets or sets the control type value. Always ControlType.CheckBox and used when adding control to its form's controls collection.
Enabled - Gets or sets a value (true/false) indicating whether the control can respond to user interaction.
Font - Gets or sets the font for the control. Refer to the Font object.
Height - Gets or sets the height of the control.
hWindow - Gets the Windows handle (hwnd) of the control.
Left - Gets or sets the distance, in pixels, between the left edge of the control and the left edge of its container's client area (normally the form).
Location - Gets or sets the top and left position of the form. Get: returns wfxPoint object. Set: (left, top).
Locked - Gets or sets a value (true/false) indicating whether the control can be moved or resized.
Parent - Gets or sets the parent container of the control.
Size - Gets or sets the size of the control. Get: returns wfxSize object. Set: (width, height).
TabIndex - Gets or sets the position that the control occupies in the TAB position.
TabStop - Gets or sets a value (true/false) indicating whether the user can use the TAB key to give focus to the control.
Tag – Gets or sets user defined text associated with the control.
Text - Gets or sets the text (caption) associated with this control.
TextAlign - Gets or sets a value indicating the alignment of the text on a control. Refer to ButtonAlignment enum.
ThreeState - Gets or sets a value (true/false) indicating whether the control will allow three check states rather than two.
Top - Gets or sets the distance, in pixels, between the top edge of the control and the top edge of its container's client area (normally the form).
UseMnemonic – Gets or sets a value (true/false) indicating whether the first character preceded by an ampersand (&) character will be used as the control's accelerator key.
Visible - Gets or sets a value (true/false) indicating whether the control is displayed.
Width - Gets or sets the width of the control.

Methods

Hide - Conceals the control from the user.
Refresh - Forces the control to invalidate its client area and immediately redraw itself.
SelectNextControl – Moves the input control to the next (True) or previous (False) control in the tab order.

SetBounds - Sets the bounds of the control to the specified location and size. (left, top, width, height).
Show – Creates and makes the control visible to the user.

Events

OnAllEvents - Special handler where all events are routed through. Use this handler if you prefer to use the Win32 api style messages and wParam and lParam parameters. Set the Handled element of EventArgs to True if you handle a message and do not want Windows to perform any further processing on the message.
OnClick - Occurs when the client area of the control is clicked.
OnDestroy - Occurs immediately before the control is about to be destroyed and all resources associated with it released.
OnMouseDoubleClick - Occurs when the control is double clicked by the mouse.
OnMouseDown - Occurs when the mouse pointer is over the control and a mouse button is pressed.
OnMouseEnter - Occurs when the mouse pointer enters the control.
OnMouseHover - Occurs when the mouse pointer rests on the control.
OnMouseLeave - Occurs when the mouse pointer leaves the control.
OnMouseMove - Occurs when the mouse pointer is moved over the control.
OnMouseUp - Occurs when the mouse pointer is over the control and a mouse button is released.