Bitmap Browser Sample Help

Sample Description: Bitmap Browser

Points of Interest

Tips and Techniques

Controls

Image
FileComboBox
FileListBox
ListBox
StopClock

For Help on Help, Press F1

Bitmap Browser

The Bitmap Browser is a finished application that can be used to quickly browse through graphic files on your system. In the sample, only Windows Bitmap (BMP) files are included, however with a few minor enhancements, this could be expanded to cover all the graphic file formats that Envelop can read. These formats are listed later in this help file.

The Bitmap Browser provides controls for selecting various Drives and Directories. This can include floppy drives and CD-ROM drives as well as hard drives. Once a Drive and Directory is selected, you can click the **Search** button. Program code will search recursively through all subdirectories finding all bitmap (BMP) files. Once a file is located, it is added to a selection list on the Bitmap Browser form. When all directories have been searched, a message box will be displayed indicating the amount of time that was taken to perform the search.

To view a particular bitmap file:

- Click on the corresponding filename in the list. A Bitmap Viewer form will appear and display the selected bitmap file.

You can change the viewing mode between full size and fit size by using the popup menu options.

Selecting a System Drive

This sample uses a **FileComboBox** named **cboSelDrive** to change drives. This control specializes in selecting Drives because the **SelType** property is set to "1 - Drives." When a specific drive is clicked, the **SelPath** property is automatically updated. Clicking on this control invokes the following Click method:

```
Sub cboSelDrive_Click()
        ' Set the current directory for the Select Directory list
        lstSelDirectory.CurrentDir = cboSelDrive.SelPath
        ' Update the Search Directory label
        lblCurDirectory.Caption = lstSelDirectory.CurrentDir
End Sub
```

This method basically sets the **CurrentDir** property of the **lstSelDirectory** (FileListBox) to the chosen drive. This automatically updates the contents of the FileListBox to include all main directories on that drive. In addition, a label on the form is updated to display the current drive and directory information.

Selecting a System Directory

This sample uses a **FileListBox** named **lstSelDirectory** to change directories. This control specializes in selecting Directories because the **ShowDirs** property is set to True. To prevent drives and files from being included, the **ShowDrives** and **ShowFiles** properties are set to False. The property **CurrentDir** displays the current working directory. The following Double-Click event is assigned to the lstSelDirectory control:

```
Sub lstSelDirectory_DblClick()
        ' Set the Select Directory current directory to the one chosen
        lstSelDirectory.CurrentDir = lstSelDirectory.SelPath

        ' Update the Search Directory label
        lblCurDirectory.Caption = lstSelDirectory.CurrentDir

        ' Update current directory of hidden file and directory listboxes
        lstHiddenFiles.CurrentDir = lstSelDirectory.CurrentDir
        lstHiddenDirs.CurrentDir = lstSelDirectory.CurrentDir
End Sub
```

As you can see by the code above, when a directory is double-clicked in the **lstSelDirectory** control, two additional FileListBox controls are updated with the **CurrentDir** information. These two controls are named: **lstHiddenFiles** and **lstHiddenDirs**. Each of these controls were embedded directly in the Property Editor. When this is done, they are not displayed on the form. The lstHiddenFiles FileListBox has its **ShowFiles** property set to True, and the lstHiddenDirs FileListBox has its **ShowDirs** property set to True as well.

This application is basically organizing the files and directories in two FileListBoxes each time a directory is clicked by the user. These two "hidden" FileListBoxes play a key role as the sub-directories are recursively searched through. It is important to note that the lstHiddenFiles control has its **Filter** property set to ".bmp." This means that only Bitmap files (.BMP) will be displayed in this list. To enhance the application to accept other graphic file formats, simply add additional formats to this Filter property. All files shown in this list will end up in the **lstBmpFiles** ListBox for user selection.

Searching a Directory for Files

The actual file search is triggered by the **Search** Button through the **btnSearch_Click** method. In this method, a Sub procedure called **GenerateBmpList** is called and given the name of the current directory.

The way this routine works is that the contents of the "hidden" list, lstHiddenDirs, is looked at line-by-line. For each sub-directory that is found, additional sub-directories are searched for. This recursive search continues down the directory structure until no additional sub-directories are found. At this point, since the lstHiddenFiles FileListBox is updated at the same time as lstHiddenDirs, all Bitmap (BMP) files would appear in this list. Each time bitmap files are present, they are extracted and added to the lstBmpFiles ListBox.

Handling Exceptions

When the **Search** Button is clicked, a recursive search operation through a Directory structure is started. Depending on the directory and the size of the disk drive, this search may take a matter of seconds. To be able to interrupt this search operation, the **Caption** on the **Clear** button is changed to "Cancel" during the search. If the "Cancel" button is clicked during the search operation, an AbortFlag property is set to True.

The GenerateBmpList Sub procedure contains the following Try/Catch code.

```
' Initiate the recursive search for matching files
Try
        GenerateBmpList lstSelDirectory.CurrentDir
Catch AbortFlag()
        InfoBox.Msg("Search operation canceled.")
End Try
```

If the Cancel button is clicked at any time during the search operation, the following program code inside the GenerateBmpList Sub procedure detects the True status of the AbortFlag and "Throws" an exception.

```
 AbortFlag = 0
' Free up CPU to update display
  Application.DoEvents()

  ' Check the AbortFlag status
  If (AbortFlag) Then Throw AbortFlag()
```

To allow for an interruption, an Application.DoEvents statement was added to the program code. This frees up the CPU and processes any outstanding events, such as a Click event from the Cancel button. If the user clicks the Cancel button before the search operation has completed, the AbortFlag property is set to True, the click is detected by the above check, and the **Throw** event is caught by the **Catch** located in the original routine. At this point, a message can be posted and the application restored to a non-search condition.

Timing Lengthy Operations

Some application functions may take a little time to complete. In the sample, depending on the directory you are searching and the size of your drive, the search may take several seconds. To show you how to time a lengthy operation, a **StopClock** object was embedded in the sample form named "**tmrStopWatch**."

When the **Search** Button is clicked, the **State** property of the tmrStopWatch is set to 3, which starts the timer running. The following options exist in the State property:

State	Action
0	Reset
1	Stopped
2	Paused
3	Running

When the search operation is complete, the State property is set to 0, which stops the timer. The total time is recorded in the **ElapsedTime** property. After the search has finished, a Message Dialog is posted indicating the total time that had elapsed during the search operation.

In addition, the total number of files that were located is determined in the following code example:

```
total_time = tmrStopWatch.ElapsedTime

file_count = lstBmpFiles.ListCount

InfoBox.Msg(file_count & " bitmap files located in " & total_time & " time.")
```

This is one way that lengthy operations may be timed. It is up to your application to determine what action you wish to carry out based on the elapsed time. In this sample, the time is merely posted in a message dialog.

Filtering File Types

In this sample, only Windows Bitmap (.BMP) files will appear in the search result list box lstBmpFiles. This is determined by the lstHiddenFiles control's **Filter** property which is set to ".bmp." This means that only Bitmap files (.BMP) will be displayed in this list.

To enhance the application to accept other graphic file formats simply add additional formats to this Filter property. All files shown in this list will end up in the **lstBmpFiles** ListBox for user selection.

Resizing Controls

The sample form contains a Resize method that is trigged each time the form is resized. The purpose of the Resize method is to allow the lstBmpFiles ListBox to grow wider and higher as the form is made wider and higher. In addition, the Directory FileListBox, the Directory Label and the two Buttons at the bottom right side of the form must also be automatically moved and resized when the form size is changed.

The **Resize** method contains program code that automatically sizes each of the necessary controls on the form. Minimum gaps between controls are defined to keep controls from overlapping each other. After all the controls are moved and resized, the Resize method invokes a **Refresh** method to redraw the form and its controls.

In the Bitmap Viewer Form, there is also a Resize method defined to keep the Image control the same size as the form. If the form is dragged to a larger size, the Image control is automatically resized in the following Resize method:

```
Sub Resize()
   ' Keep the size of the image control the same as the form
   imgViewer.Left = 0
   imgViewer.Top = 0
   imgViewer.Width = BrowserDisplayForm.ScaleWidth
   imgViewer.Height = BrowserDisplayForm.ScaleHeight

   ' Let's update the display
   imgViewer.Refresh
End Sub
```

These are just a few examples of how a Resize method can be used to automatically resize and locate controls on the form.

Disabling Controls during Lengthy Operations

When an application operation takes a few seconds to complete, it is a good idea to disable controls on the form to prevent unwanted clicks while an operation, in this case the search operation, is running. Various controls on the form can be disabled by setting their **Enabled** property to False during the search operation, and then setting the property back to True once the search is completed. Below is an example of how to disable controls on the form:

```
lstBmpFiles.Enabled = False
lstSelDirectory.Enabled = False
cboSelDrive.Enabled = False
btnClear.Caption = "Cancel"
btnSearch.Enabled = "False"
```

Notice that setting the Enabled property to False will disable the control. Likewise, setting the Enabled property to "True" will enable or activate the control. Another way to let the user know that a lengthy operation is underway is to change the Mouse cursor to an "Hour glass" during the operation, then reset it to the standard "Arrow pointer" when the task is finished.

Viewing a Bitmap at Full Scale

In this Bitmap Browser sample, if you click on one of the files displayed in the search result ListBox, a secondary form will be displayed on the screen. This form can be used to view the Bitmap file. The Bitmap Viewer form has a menu option called "Full" that sets the scale factor to 1:1. This function is done in the following method:

```
Sub ScaleFull_Click()
   ' View the bitmap at a 1:1 scale
   imgViewer.ScaleX = 1
   imgViewer.ScaleY = 1
   imgViewer.Refresh

   ' Set the resize mode to clip
   imgViewer.ResizeMode = 1

   ' Clear exiting checkmarks
   ClearMenuCheckMarks
   ' Add a checkmark to the Fit entry
   BrowserDisplayViewMenu.CheckItem("ScaleFull", 1)
End Sub
```

As you examine this program code, that is attached to the Menu/Full Click event, notice that the **ScaleX** and **ScaleY** properties of the Image control imgViewer are set to 1. In addition, be sure to notice the **Resize** mode setting, which is 1. This allows the size of the Bitmap Viewer form to be changed while leaving the image at a full 1:1 scale.

This code also automatically "checks" the Full menu entry to indicate the current view scale.

Fitting a Bitmap to an Image Control

In this Bitmap Browser sample, if you click on one of the files displayed in the search result ListBox, a secondary form will be displayed on the screen and can be used to view the Bitmap file. The Bitmap Viewer form has a menu option "Fit," which fits the bitmap to the Image. Since the Image control is the same size as the Form, the bitmap is automatically fit to the boundaries of the form. This function is performed in the following method:

```
Sub ScaleFit_Click()
  ' View the bitmap at a fit mode
  imgViewer.CropXOffset = 0
  imgViewer.CropYOffset = 0
  imgViewer.CropXSize = BitmapFile.Width
  imgViewer.CropYSize = BitmapFile.Height
  imgViewer.Refresh

  ' Set the resize mode to fit
  imgViewer.ResizeMode = 0

  ' Clear exiting checkmarks
  ClearMenuCheckMarks
  ' Add a checkmark to the Fit entry
  BrowserDisplayViewMenu.CheckItem("ScaleFit", 1)
End Sub
```

As you examine this program code that is attached to the Menu/Fit Click event, notice that the **CropXOffset** and **CropYOffset** properties of the Image control imgViewer are set to 0. In addition, notice that the **Resize** mode is set to 0. This allows changes to the size of the Bitmap Viewer form and automatically resized the image to fit the form.

This code also automatically "checks" Fit menu entry to indicate the current view scale.

Stripping Brackets from Directory Strings

As the program code browses through the directory structure searching for files, it presents the names of all the directories presented in the lstHiddenDirs with brackets [] around them. Since the program code only wants to use the names of the directories, the following function is used to strip the brackets from directory name:

```
Function StripBrackets(dir_name As String) As String
    Dim left_char As String
    Dim right_char As String
    Dim string_size As Integer

    ' Save the first character
    left_char = Left(dir_name, 1)
    ' Save the last character
    right_char = Right(dir_name, 1)

    ' If brackets enclose the string, remove them
    If left_char == "[" And right_char == "]" Then
        string_size = Len(dir_name) - 2
        StripBrackets = Mid(dir_name, 2, string_size)
    Else
        ' Pass back the original string since it has no brackets
        StripBrackets = dir_name
    End If

End Function
```

This is just a simple technique that uses the **Mid** and **Len** language commands to remove unwanted characters from a text string.

Interchanging Information between Forms

One of the strongest features of Envelop is the ease in which Forms may interact with each other. For example, from one form, you may directly set another Form's property, execute one of its Methods or invoke one of its Events. The following program code is the Click event that is associated with the ListBox containing the list of bitmap files that were found in the search operation.

```
Sub lstBmpFiles_Click()
    Dim option As long
    Dim result As long
    Dim bmp_file As String

    ' Set a variable to be the name of sound file including absolute path
    bmp_file = lstBmpFiles.ItemString(lstBmpFiles.ListIndex)

    ' Display the selected bmp file
    BrowserDisplayForm.BitmapFile.FileName = bmp_file
    BrowserDisplayForm.imgViewer.Refresh

    If BrowserDisplayForm.Visible == 0 Then
      BrowserDisplayForm.Show
    End If

    ' Update the correct display mode
    BrowserDisplayForm.UpdateDisplay

End Sub
```

In this Click method, the **FileName** property of the Bitmap object is being set, the **Refresh** method is being executed, the **Show** method is being executed, and finally the **UpdateDisplay** method is executed. The UpdateDisplay method is a custom method on the Bitmap Viewer Form that determines which scale factors to apply to the image.

Supported Graphic File Formats

Envelop's Bitmap object supports the following file formats:

Number	Bitmap Format
0	TIF_UNCOMPRESSED
1	BMP_UNCOMPRESSED
2	PCX
3	TARGA
4	GIF
5	WPG
6	WMF
7	TIFF_HUFFMAN
8	TIFF_G3_FAX
9	TIFF_LZW
10	TIFF_G4_FAX
11	DCX
12	BMP_COMPRESSED
13	JPEG
14	EPS
15	PICT
16	TIFF_PACK
17	TIFF_2D
18	CALS
19	LASER_DATA
20	XBM
21	MACPAINT
22	GX2
23	KOFAX
24	IOCA
25	ICON
26	IFF_ILBM
27	CLIP
28	IMG
29	BROOK_TROUT
30	MSP
31	CUT
32	TARGA16
33	CCITT_G3
34	CCITT_G4
35	XPM
36	XWD
37	RAST
38	ASCII
39	PHOTOCD
40	TIFF JPEG
41	PHOTOSHOP