TBrowseFolder Component

Written by Todd Fast

Version 1.0, lego 9-25-96

Description:

Native Delphi component that encapsulates the SHBrowseForFolder interface, which allows Win32 users to select a directory using the standard Explorer-like treeview dialog.

Contact:

tfast@eden.com
pencilneck@hotmail.com

Distribution:

This component is freeware. As such, Pencilneck Software gives no warranty to its accuracy, fitness for any particular use, effects of use, or reliability. This component may not be distributed as a part of another component package without Pencilneck Software's written consent. It may be freely distributed, although it must be distributed with all original files in their original format intact. If you use this component in your software, please include an acknowledgment that portions are copyrighted by Pencilneck Software. Please contact the author, Todd Fast, at one of the above addresses with questions, comments, bug-reports or any updates you make to the component.

Properties:

CallbackParam:
  App-specific value passed to the callback function from the browse dialog.
Flags:
  Set of flags for determining what the browse dialog will allow the user to choose. Enforces these restrictions by enabling or disabling the OK button when a user chooses a particular type of file item.
Folder:
  The top-level folder displayed in the browse dialog. foDesktop is the default and is what users are used to seeing in Explorer.
ShowFullPath:
  Enables or disables a custom feature that shows the selected path in the status area of the browse dialog. Must have the bfStatusText flag set.
Title:
  The title text shown in the browse dialog.
DisplayName:
  Read-only. The display name returned from the dialog in the BROWSEINFO structure;
ImageIndex:
  Read-only. The selected item's image index in the system image list. Returned in the BROWSEINFO structure.
Directory:
  Read-only. The path of the chosen directory.
BrowseDialogShowing:
  Read-only. Set to TRUE when the browse dialog is already showing.

Methods:

constructor Create(AOwner: TComponent);

Standard constructor for TComponent.

function Execute: Boolean;

Shows the browse dialog and allows the user to choose a directory. Returns true if the user chose the OK button, FALSE if he or she chose the Cancel button.

procedure SetStatusText(const Hwnd: HWND; const StatusText: String);

Hwnd:

Handle of the browse dialog.

StatusText:

The text message.

Sets the status area text to the text message. You must have the bfStatusText flag set to see the status text.

procedure SetSelectionPIDL(const Hwnd: HWND; const ItemIDList: PItemIDList);

Hwnd:

Handle of the browse dialog.

ItemIDList:

Pointer to an item identifier list, also know as a 'pidl', which identifies a folder.

Sets the selection in the browse dialog to the folder represented by the pidl. The pidl is an opaque binary value and would need to be created by some other Shell API like SHGetSpecialFolderLocation. Don't forget to deallocate and pidl you obtain yourself with the CoTaskMemFree function or equivalent. Not generally as useful as the next method.

procedure SetSelectionPath(const Hwnd: HWND; const Path: String);

Hwnd:

Handle of the browse dialog.

Path:

String value of the path to select.

Sets the selection in the browse dialog to the folder in the Path parameter. The path can be in long or 8.3 format.

procedure EnableOK(const Hwnd: HWND; const Value: Boolean);

Hwnd:

Handle of the browse dialog.

Value:

Desired state of the browse dialog OK button.

Sets the enabled state of the OK button in the browse dialog. Note: You can use this to override the restrictions set in the Flags property, but if the user selects an item that the Flags item restricts, the returned directory will be an empty string.

Events:

OnInitialized(Hwnd: HWND; CallbackParam: LPARAM)

Hwnd:

Handle of the browse dialog.

CallbackParam:

Value of the CallbackParam property of the TBrowseFolder object whose Execute procedure was called.

Fired when the browse dialog is done initializing.

OnSelectionChanged(Hwnd: HWND; CallbackParam: LPARAM;
const ItemIDList: PItemIDList)

Hwnd:

Handle of the browse dialog.

CallbackParam:

Value of the CallbackParam property of the TBrowseFolder object whose Execute procedure was called.

ItemPidl:

Pidl of the selected item. See the note below for info on getting the directory path from the pidl.

Fired when a new folder in the browse dialog is selected.

Comments:

This component wraps a few of the Win32 shell functions to display the Windows standard folder browse dialog. Frankly, I hacked this component together over the course of a couple of evenings based on Microsoft's sketchy documentation and some of their C header files, so I can't vouch for the complete accuracy of the component. It does seem to work quite well, though, and I haven't experienced any problems with it, so I tend to think everything works smoothly. I also added to the basic functionality of showing the browse dialog the capability to show only particular directories, based on the handy SHGetSpecialFolderLocation function. Some of the folder locations might not be defined on your system, so you may want to only use the common folders like foDesktop or foNetwork in your software. If anyone has some comments on how I implemented the component (or simply knows better), please email me and correct any errors I've made. Please forgive me for documenting the component in the source file instead of generating a help file; I've tried to make up for it by extensively commenting the code.

Hints:

- If you want to get the selected directory path in the OnSelectionChanged event, use the SHGetPathFromIDList function on the ItemPidl parameter passed into the event handler.
- The SetStatusText, SetSelectionPIDL, SetSelectionPath, and EnableOK methods ecapsulate the messages you can send to the browse dialog while it is active. Use these functions from within the TBrowseFolder event handlers to make changes to the browse dialog instead of using SendMessage (although that would be perfectly acceptable, and this file defines all the constants you would need.)
- Use the SHGetFileInfo function to retrieve extended information about the selected folder.
- For more information, lookup SHBrowseForFolder, BROWSEINFO, and BrowseCallbackProc in the Win32 online help.
- Beware! The Microsoft documentation on these functions shipped with Delphi is not entirely accurate. In most cases, they've reversed the location of certain parameters sent to the callback function or of messages you can send to the browse dialog. Compare my implementation below with the documentation for more information.