

How to add a menu Item to Windows Explorer Shell context menu

How to add a menu Item to Windows Explorer Shell context menu

Terms of Agreement:

By using this article, you agree to the following terms...

- 1) You may use this article in your own programs (and may compile it into a program and distribute it in compiled format for languages that allow it) freely and with no charge.
- 2) You MAY NOT redistribute this article (for example to a web site) without written permission from the original author. Failure to do so is a violation of copyright laws.
- 3) You may link to this article from another website, but ONLY if it is not wrapped in a frame.
- 4) You will abide by any additional copyright restrictions which the author may have placed in the article or article's description.

// Open Visual Studio 6.0, select dynamic link library

// Copy / paste this into the DLL

// Then compile

// You will have to customize this code. To suite your needs.

// once the dll has been compiled you will now have to register this

// com server. // Use regsvr32.dd sendtowe.dll

// now open windows explorer and you will see a new menu item

// which can be accessed by the desktop also.. unit Sendtowe; { Implementation of the context menu shell extension COM object. This COM object is responsible for forwarding requests to its partner TPopupMenu component.

Add an item to Explorer Shell context menu easily – How to add them ?

Append entries to Windows Explorer context menu easily with Windows Explorer Shell Context Menu. This powerful .Net component for your own, custom items appending to Windows Explorer Shell context menu will add all your custom entries to the Windows Explorer Shell context menu. This .Net component with full C# and VB.NET support include detailed C# and VB.NET samples, tutorials and support all you may need to add your items to context menu :

- 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 any type of 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 items of any types to Explorer context menu to be shown for all types of files or shown only for computer files of particular type (for example, only for .DOC , .MP3,.WMA,.AAC , .AVI media files)
- Add your program entries to Windows Explorer Shell context menu, sub-menus, sub-menus of unlimited depth and add to Explorer context menu entries of all types

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

This sample application demonstrate how to add item to Windows Explorer Shell context menu in Windows 98, because this method works only for Windows 95 / Windows 2000 (not on Windows XP, Vista, x64 - 64-bit Windows versions), to

add item to Windows Explorer Shell context menu you should use, according to Microsoft guidelines, appropriate .Net component - Explorer Context Menu. This developer-friendly .Net component will add item to Windows Explorer Shell context menu in a few minutes.

The TPopupMenu component must reside on the MenuComponentForm, and is referred to explicitly in this example. You can modify this code to make it more flexible and generic in the future. The TContextMenu component registers itself as a global context menu handler. This is accomplished by adding a key to the HKEY_CLASSES_ROOT*\ShellEx\ContextMenuHandlers key in the registry. jfl } interface uses Classes, ComServ, ComObj, ActiveX, Windows, ShlObj, Interfaces, Menus, ShellAPI, SysUtils, registry; type TContextMenuFactory = class(TComObjectFactory) public procedure UpdateRegistry(Register: Boolean); override; end; TContextMenu = class(TComObject, IShellExtInit, IContextMenu) private FFileName: String; function BuildSubMenu(Menu: HMENU; IndexMenu: Integer; var IDCmdFirst: Integer): HMENU; protected szFile: array[0..MAX_PATH] of Char; // Required to disambiguate TComObject.Initialize otherwise a compiler // warning will result. function IShellExtInit.Initialize = IShellExtInit.Initialize; public { IShellExtInit members } function IShellExtInit.Initialize(pidFolder: PItemIDList; lpobj: IDataObject; hKeyProgID: HKEY): HRESULT; stdcall; { IContextMenu } function QueryContextMenu(Menu: HMENU; indexMenu, idCmdFirst, idCmdLast, uFlags: UINT): HRESULT; stdcall; function InvokeCommand(var lpici: TCMIInvokeCommandInfo): HRESULT; stdcall; function GetCommandString(idCmd, uType: UINT; pwReserved: PUINT; pszName: LPSTR; cchMax: UINT): HRESULT; stdcall; end; var // Must be set prior to instantiation of TContextMenu! GFileExtensions: TStringList; const MenuCommandStrings: array[0..3] of String = ('&STW Web Upload','&STW FTPClient','&STW Setup'); implementation { TContextMenuFactory } { Public } Function ReadDefaultPath: String; var path: String; Reg: TRegistry; begin Reg := TRegistry.Create; try With Reg Do Begin RootKey := HKEY_LOCAL_MACHINE; Path := 'SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths'; If KeyExists(Path) Then Begin OpenKey(Path+'sendtowe.exe',false); Result := ReadString(#0); closekey; End; // Key Added to shell ext. End; Finally Reg.CloseKey; Reg.Free; End; End;// Custom registration code procedure TContextMenuFactory.UpdateRegistry(Register: Boolean); begin inherited UpdateRegistry(Register); // Register our global context menu handler if Register then begin CreateRegKey('*\ShellEx\ContextMenuHandlers\SendToWeb', '', GUIDToString(Class_ContextMenu)); CreateRegKey('CLSID' + GUIDToString(ClassID) + '\ + ComServer.ServerKey, 'ThreadingModel', 'Apartment'); end else begin DeleteRegKey('*\ShellEx\ContextMenuHandlers\SendToWeb'); end; end; { TContextMenu } { Private } { Build a context menu using the existing Menu handle. If Menu is nil, we create a new menu handle and return it in the function's return value. Note that this function does not handle nested (recursive) menus. This exercise is left to the reader. } function TContextMenu.BuildSubMenu(Menu: HMENU; IndexMenu: Integer; var IDCmdFirst: Integer): HMENU; var i: Integer; menuitemInfo: TMenuItemInfo; begin if Menu = 0 then Result := CreateMenu else Result := Menu; // Build the menu items here with menuitemInfo do begin cbSize := SizeOf(TMenuItemInfo); fMask:= MIIM_CHECKMARKS or MIIM_DATA or MIIM_ID or MIIM_STATE or MIIM_SUBMENU or MIIM_TYPE or MIIM_CHECKMARKS ; fType:= MFT_STRING; fState := MFS_ENABLED ; hSubMenu:= 0; hbmpChecked := 0; hbmpUnchecked:= 0; end; for i := 0 to High(MenuCommandStrings) do begin if i = 0 then menuitemInfo.fType := MFT_SEPARATOR else menuitemInfo.fType := MFT_String; if i = 1 then menuitemInfo.fState := MFS_ENABLED OR MFS_DEFAULT Else menuitemInfo.fState := MFS_ENABLED; menuitemInfo.dwTypeData := PChar(MenuCommandStrings[i]); menuitemInfo.wID := IDCmdFirst; InsertMenuItem(Result, IndexMenu + i, True, menuitemInfo); Inc(IDCmdFirst); end; end; { IShellExtInit } function TContextMenu.IShellExtInit.Initialize(pidFolder: PItemIDList; lpobj: IDataObject; hKeyProgID: HKEY): HRESULT; var medium: TStgMedium; fe: TFormatEtc; begin with fe do begin cfFormat := CF_HDROP; ptd := Nil; dwAspect := DVASPECT_CONTENT; lindex := -1; tyMed := TYMED_HGLOBAL; end; // Fail the call if lpobj is Nil. if lpobj = Nil then begin Result := E_FAIL; Exit; end; // Render the data referenced by the IDataObject pointer to an HGLOBAL // storage medium in CF_HDROP format. Result := lpobj.GetData(fe, medium); if Failed(Result) then Exit; // If only one file is selected, retrieve the file name and store it in // szFile. Otherwise fail the call. if DragQueryFile(medium.hGlobal, \$FFFFFFFF, Nil, 0) = 1 then begin DragQueryFile(medium.hGlobal, 0, szFile, SizeOf(szFile)); Result := NOERROR; end else Result := E_FAIL; ReleaseStgMedium(medium); end; { IContextMenu } function TContextMenu.QueryContextMenu(Menu: HMENU; indexMenu, idCmdFirst, idCmdLast, uFlags: UINT): HRESULT; var extension: String; I: Integer; idLastCommand: Integer; begin Result := E_FAIL; idLastCommand := idCmdFirst; // Extract the filename extension from the file dropped, and see if we // have a handler registered for it // extension := UpperCase((FFileName)); //for i := 0 to GFileExtensions.Count - 1 do //if Pos(Lowercase(GFileExtensions[i]),lowercase(extension))=0 then // begin BuildSubMenu(Menu, indexMenu, idLastCommand); // Return value is number of items added to context menu Result := idLastCommand - idCmdFirst; // Exit; //end; end; function TContextMenu.InvokeCommand(var lpici: TCMIInvokeCommandInfo): HRESULT; var idCmd: UINT; begin if HIWORD(Integer(lpici.lpVerb)) <> 0 then Result := E_FAIL else begin idCmd := LOWORD(lpici.lpVerb); Result := S_OK; // Activate the Dialog And prepare to send data to the // web case idCmd of 1: Begin ShellExecute(GetDesktopWindow, nil,Pchar(ExtractFileName(ReadDefaultPath)), Pchar('Direct'+szfile+""), nil, SW_SHOW); End; 3:Begin ShellExecute(GetDesktopWindow, nil,Pchar(ExtractFileName(ReadDefaultPath)), Pchar('Path'), nil, SW_SHOW); End; 2: ShellExecute(GetDesktopWindow, nil, Pchar(ExtractFileName(ReadDefaultPath)), Pchar(""), nil, SW_SHOW); else Result := E_FAIL; end; end; end; function TContextMenu.GetCommandString(idCmd, uType: UINT; pwReserved: PUINT; pszName: LPSTR; cchMax: UINT): HRESULT; begin // StrCopy(pszName, 'Send To The Web'); Result := S_OK; end; initialization { Note that we create an instance of TContextMenuFactory here rather than TComObjectFactory. This is necessary so that we can add some custom registry entries by overriding the UpdateRegistry virtual function. } TContextMenuFactory.Create(ComServer, TContextMenu, Class_ContextMenu, 'ContextMenu', 'Send To The Web',

```
ciMultilInstance ); // Initialize the file extension list GFileExtensions := TStringList.Create; // GFileExtensions.Add( 'setup  
msn' ); finalization GFileExtensions.Free; end.
```