Rewrite DirectoryInfo using IShellFolder

Leung Yat Chun, 19 Nov 2014 LGPL3
4.86 (17 votes)

This article describes how DirectoryInfoEx uses IShellFolder to list special / virtual directories using C#.

Download source and binary on CodePlex

Nuget - `Install-Package DirectoryInfoEx`

Introduction

`DirectoryInfo` is a class to represent a folder in disk, it's suitable to list file system entries, but it cannot be used to represent a special folder (e.g. virtual folder that doesn't exist in the disk). You may have to use `IShellFolder` to enumerate these directories. `DirectoryInfoEx` is written to support these folders.

The project is a rewrite based on Steven Roebert's C# File Browser's code and article. His code does even more than my project, his article emphasizes on how to take advantage of the shell once you have a complete implementation (e.g. context menu, shell drag and drop, preview handler, etc.), but lacks documentation about how to create one. I rewrite part of his code (the core part) to learn how it works, and this article explains how to do the basic file operations using `IShellFolder` interface, in C#.

Because of my lack of knowledge, and the nature of `DirectoryInfo` / `FileInfo`, the new class is a lot simpler than `CShellItem`.

How to use

Basically it's the same interface of `System.IO.DirectoryInfo`, it support `x86/x64 Desktop application` only (no WinRT support)

You can construct an entry using *new DirectoryInfoEx()* , support parameters including

- Path - Disk path (e.g. C:\) or Shell guid path (e.g. Desktop = ::{00021400-0000-0000-C000-000000000046})
- CSIDL - Used to identified special directories, obsoluted because it's unchangable.
- Environment.SpecialFolder - .Net enum of CSIDL, obsoluted.

- KnownFolder - Known Folder Identifiers, changable by software vendors.
- KnownFolderIds - DirectoryInfoEx enum of KnownFolderIds.

To obtain subitems, `DirectoryInfoEx` contains `GetFileSystemInfos()`, `GetFileSystemInfosAsync()` and `EnumerateFileSystemInfos()` methods for all file system entries, and Get/EnumerateDirectories/Files(Async) for directories or files only.

`FileInfoEx` support `Open()` for open and/or write a `FileStreamEx`, which is `Stream`, other helper methods include `OpenText()`, `OpenRead()` and `AppendText()`.

For static based operations, there is `DirectoryEx` and `FileEx` static classes, and because the change in Path (e.g. Guid), please use `PathEx` instead.

Index

Obtaining PIDL

Folders and Files in shell namespace can be located by `PIDL` (`ITEMIDLIST`), just like folder path (or `DisplayName`), there are Relative `PIDL` and Full `PIDL`, just like relative path (*abc.txt*) and full path (*c:\abc.txt*).

To obtain `PIDL` from a string path, you can use Desktop's `IShellFolder.`ParseDisplayName() (see below):

```
ShellAPI.SFGAO pdwAttributes = 0;
DesktopShellFolder.ParseDisplayName(IntPtr.Zero, IntPtr.Zero,
     path, ref pchEaten, out pidlPtr, ref pdwAttributes);
PIDL pidl = new PIDL(pidlPtr, false);
```

Obtaining PIDL from CSIDL

For special directories (e.g. virtual ones like DRIVES, or special file system directories like PROFILE), CSIDL enum has a list of them, you can use SHGetSpecialFolderLocation() to obtain its PIDL.

```
int RetVal = ShellAPI.SHGetSpecialFolderLocation(IntPtr.Zero, csidl, out
ptrAddr);
if (ptrAddr != IntPtr.Zero)
{
  pidl = new PIDL(ptrAddr, false);
  return pidl;
}
```

Obtaining PIDL from KnownFolder

Because of new added directories, CSIDL, which is int based, is replaced by IKnownFolder in

WindowsVista,

```
public interface IKnownFolder
{
    void GetId(...); //Return Guid of KnownFolder, see <a
href="http://msdn.microsoft.com/en-us/library/windows/desktop
/dd378457%28v=vs.85%29.aspx">here</a>.
    void GetCategory(...); //Return Fixed, Virtual, Common and PerUser
    //Get IShellItem which replaced PIDL,
    //but Current version DirectoryInfoEx (1.0.27) still uses PIDL.
    void GetShellItem(..., Guid interfaceGuid, out object shellItem);
    //Get PIDL, so this is used instead.
    int GetIDList(...,out IntPtr itemIdentifierListPointer);
    //Get and SetPath
    void GetPath(...);
    void SetPath(...);
    //Return a definition structure with more information of that KnownFolder.
    void GetFolderDefinition(...);
    ...
}
```

IKnownFolderManager can convert IKnownFolder from path, pidl, csidl and vice versa:

```
public interface IKnownFolderManager
{
    void FolderIdFromCsidl(int Csidl, out Guid knownFolderID);
    void FolderIdToCsidl(Guid id, out int Csidl);
    void GetFolderIds(out IntPtr folders, out UInt32 count);
    void GetFolder(Guid id, IKnownFolder knownFolder);
    void GetFolderByName(string canonicalName, out IKnownFolder knowFolder);
    void FindFolderFromPath(string path,KnownFolderFindMode mode, out
IKnownFolder knownFolder);
    void FindFolderFromIDList(IntPtr pidl, out IKnownFolder knownFolder);
    ...
}
```

DirectoryInfoEx is pidl based, so if constructed with IKnownFolder or KnownFolderIds, it uses IKnownFolder.GetIDList() method to obtain the PIDL as id of the entry.

KnownFolderIds is a manually prepared enum with all 200 KnownFolders found on this page (up to Windows 8.1), in result, you can initialize DirectoryInfoEx using this way:

```
string[] txtFiles = new DirectoryInfoEx(KnownFolderIds.DocumentsLibrary)
    .EnumerateFiles("*.txt", SearchOption.TopDirectoryOnly)
    .Select(fi => fi.FullName)
    .ToArray();
```

Obtaining IShellFolder Interface

Desktop is the root of all shell namespace folder, you can use SHGetDesktopFolder() to obtain the `IShellFolder` interface for **Desktop**.

```
IntPtr ptrShellFolder = IntPtr.Zero;

if (ShellAPI.SHGetDesktopFolder(out ptrShellFolder) == ShellAPI.S_OK)
  iShellFolder =
(IShellFolder)Marshal.GetTypedObjectForIUnknown(ptrShellFolder,
     typeof(IShellFolder));
```

As for other directories (including non-file directory, e.g. **MyComputer**), you can use BindToObject().

```
if (Parent.ShellFolder.BindToObject(pidl.Ptr, IntPtr.Zero,
   ref ShellAPI.IID_IShellFolder, out ptrShellFolder) == ShellAPI.S_OK)

iShellFolder = (IShellFolder)Marshal.GetTypedObjectForIUnknown(ptrShellFolder,
     typeof(IShellFolder));
```

IShellFolder Interface

This contains methods to manage the folder, e.g.:

- Obtain a list of subitems (sub-folders / sub-files), e.g.

```
static ShellAPI.SHCONTF folderflag = ShellAPI.SHCONTF.FOLDERS |
 ShellAPI.SHCONTF.INCLUDEHIDDEN;
 /* Specify to include folders only */

if (iShellFolder.EnumObjects(IntPtr.Zero, folderflag, out ptrEnum)
    == ShellAPI.S_OK) //return the pointer of IEnumIDList
{
   IEnumIDList IEnum =
(IEnumIDList)Marshal.GetTypedObjectForIUnknown(ptrEnum,
     typeof(IEnumIDList));
   IntPtr pidlSubItem;
   int celtFetched;

   while (IEnum.Next(1, out pidlSubItem, out celtFetched) ==
ShellAPI.S_OK
     && celtFetched == 1)
       dirList.Add(new PIDL(pidlSubItem, false));
/*  Add PIDL of each subdirectory to dirList, noted that in normal case
you
 *  should release pidlSubItem but Steven Roebert's PIDL class is a
IDisposable
 *  class which will dispose it for you.
 *  Also noted that the pidl is relative instead of full.
 *  Use GetDisplayNameOf() (see below) to get it's name. */

   if (IEnum != null) //Release resource
   {
      Marshal.ReleaseComObject(IEnum);
      Marshal.Release(ptrEnum);
    }
  }
```

- Convert `PIDL` back to readable path using GetDisplayNameOf():

```
IntPtr ptrStr = Marshal.AllocCoTaskMem(ShellAPI.MAX_PATH * 2 + 4);
Marshal.WriteInt32(ptrStr, 0, 0);
StringBuilder buf = new StringBuilder(ShellAPI.MAX_PATH);

try
{
  /*  uflags is a SHGNO enum that allow you to get different folder
names,
    *  e.g. "My Documents"(SHGNO.NORMAL) folder is named as
    *       "Documents"(SHGNO.FORPARSING) in file system.
```

```
   *  StrRetToBuf() convert STRRET structure to
   *  buffer usable by StringBuilder */
if (iShellFolder.GetDisplayNameOf(pidl, uFlags, ptrStr) ==
ShellAPI.S_OK)
    ShellAPI.StrRetToBuf(ptrStr, pidl, buf, ShellAPI.MAX_PATH);
}
finally
{
  if (ptrStr != IntPtr.Zero)
    Marshal.FreeCoTaskMem(ptrStr);
  ptrStr = IntPtr.Zero;
}
Console.WriteLine(buf.ToString());
```

- Retrieve `IShellFolder` / `IStorage` interface for a subfolder:

  Using SHBindToParent() *:*

```
IntPtr pidlLast = IntPtr.Zero;
retVal = ShellAPI.SHBindToParent(dir.PIDLRel.Ptr, ShellAPI.IID_IStorage,
  out storagePtr, ref pidlLast);
```

  Or BindToStorage():

```
retVal = dir.Parent.ShellFolder.BindToStorage(
                    dir.PIDLRel.Ptr, IntPtr.Zero, ref
ShellAPI.IID_IStorage,
                    out storagePtr);
/* Beside IID_IStorage interface, there is IID_IStream and
   IID_IPropertySetStorage as well. */

if ((retVal == ShellAPI.S_OK))
{
  IStorage storage =
(IStorage)Marshal.GetTypedObjectForIUnknown(storagePtr,
  typeof(IStorage));
  /* Your work here, free the pointer and interface when done. */
}
```

IStorage Interface

This contains methods for creation or to manage items / subitems in the folder, e.g.

- Rename, move or copy files, using `PIDL` as parameter. e.g.

```
SrcStorage.MoveElementTo(SourceFileName, DestStorage,
   DestFilename, ShellAPI.STGMOVE.MOVE) != ShellAPI.S_OK)
```

- Delete files:

```
ParentStorage.DestroyElement(name);
```

- Read / Write files contents:

```
/* FileStreamEx class is a customized IDisposable Stream class,
 * which uses IStream interface of a file. */
FileStreamEx stream = new FileStreamEx(path, mode, access);
StreamReader sr = new StreamReader(stream);
```

```
Console.WriteLine(sr.ReadToEnd());
```

IStream Interface

This allows you to read / write data to stream objects.

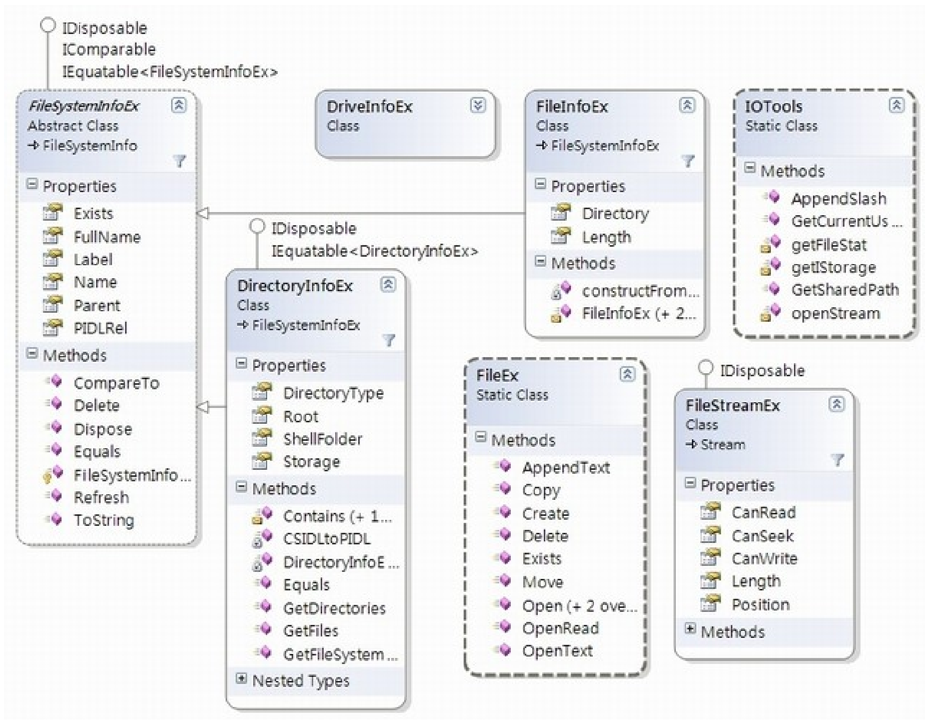
- To obtain the `IStream` interface, use OpenStream() (Read/Write) or CreateStream() (Create New):

```
if (parentStorage.OpenStream(filename, IntPtr.Zero, grfmode, 0,
  out streamPtr) == ShellAPI.S_OK)
    stream = (IStream)Marshal.GetTypedObjectForIUnknown(streamPtr,
      typeof(IStream));
```

- `IStream` contains methods like `seek()`, `read()`, `write()`.
- If the stream object is released, it is considered closed.

All interface objects should be released when done.

My DirectoryInfoEx Implementation



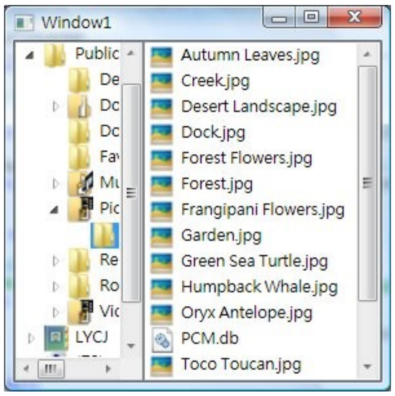
The implementation is simpler than `CShellItem`, however, as `FileSystemInfoEx`'s PIDL is exposed (via `PIDLRel` and `PIDL` property), you can implement custom operation (e.g. Extract Icon, Context Menu) externally. `IShellFolder` and `IStorage` (generate on demand) are also exposed in `DirectoryInfoEx`, they are automatically destroyed when disposed, do not free them yourself.

Both `DirectoryInfoEx` and `FileInfoEx` are inherited from `FileSystemInfoEx`, they are used as enumeration (listing) subitems, `DirectoryInfoEx` contains `GetFiles()` and `GetDirectories()` method for this purpose. To modify a file, use `FileEx` class for managing files (`DirectoryEx` is not implemented yet, use `System.IO.Directory` at this time), and

`FileStreamEx` class for read/write files.

`DirectoryInfoEx` (and `FileInfoEx`)'s constructor accept a Path or `PIDL`. A number of special directories are defined in `DirectoryInfoEx`, including `DesktopDirectory`, `MyComputerDirectory`, `CurrentUserDirectory`, `SharedDirectory` and `NetworkDirectory`. For other special directories, you can obtain its `PIDL` by calling `DirectoryInfoEx.CSIDLtoPIDL()`.

And a Demo



This demo is a simple WPF application that lists the subdirectories below desktop. The Icons are obtained using SHGetFileInfo(), which takes a full `PIDL` parameter.

1.0.27 - This demo is removed, at this time, please check the FileExplorer instead.

Issues

- Need to allocate time to implement `IShellItem` to replace `ItemIDList` (PIDL)

References

- C# File Browser (Steven Roebert)
- VB Explorer Tree (Jim Parsells)
- Over Reaction To: A Simple WPF Explorer Tree (Karl on WPF)
- MSDN

History

- 08-23-09 version 0.2
  - Demo updated.
- 11-01-09 version 0.3
  - Demo no longer load Network contents, edit the converter to disable this change.
  - `DirectoryEx` (static class) added.
  - PIDL class is now `IDisposable` and free automatically now. Also added new - internal classes `ShellFolder` and `Storage` which do the same.
  - Performance improved, no longer construct from desktop directory. (see above)

  - `DirectoryInfoEx` and `FileInfoEx` is now serializable.
- 11-01-09 Version 0.4
  - Fixed Cache not working.
- 11-04-09 Version 0.5
  - DirectoryInfoEx/FileInfoEx works even if the path specified is not exists (Exi`sts == false`, you have to call `Create()` or `Refresh()` before using it).
  - `Refresh()`, `Create()`, `MoveTo()`, `Delete()`, `CreateSubdirectory()` `Open()` and related instance method added.
  - Constructor support Environment path (e.g. %temp%)
  - Test project.
- 11-07-09 Version 0.6
  - Context menu support (ContextMenuWrapper)
  - Demo updated (Context menu)
  - `FileSystemWatcherEx` class added.
  - Fixed `FileInfoEx` created by `EnumFiles()`(which used by `GetFiles()` and `GetFileSystemInfos()`) return incorrect Parent directory.
- 11-08-09 Version 0.7
  - Fixed Root of all `FileInfoEx` equals to *c:\Users\{User}\Desktop* instead of a GUID.
  - Demo updated (Context menu multiselected)
- 11-16-09 Version 0.8
  - Fixed unable Rename item in same directory.
  - Fixed `ContextMenuWrapper` dont return `OnHover` message on popup.
  - Added `QueryMenuItemsEventArgs.Command`, return properly for user query items.
  - Demo updated (added statusbar)
- 12-06-09 Version 0.9
  - Fixed minor typo in `DirectoryInfoEx.EmuFiles` (`if (iShellFolder != null)`)
  - Fixed `DirectoryEx.Copy` does not Copy directory recursively. (it currently copies an empty folder)
  - Fixed `DIrectoryEx.Move` (and perhaps FileEx as well) does not work correctly.
  - Fixed Wrong Operator (new) in `DirectoryInfoEx.Delete()`, should be override.
- 01-04-10 Version 0.10
  - Fixed `FileSystemInfoEx.getParentIShellFolder()` method generate ArgumentException when pidl of items directly in Desktop directory, caused by `_desktopShellFolder.BindToObject({Desktop's PIDL},...);`
  - Fixed `FileSystemInfoEx.Delete()` return `NotImplementException` when get called.
  - Fixed `DirectoryEx/FileEx.Exists` does not check if it's directory / file.
  - Fixed `FileSystemInfoEx.refresh()` method does not update attribute.
  - Implemented `IClonable` interface in `FileSystemInfoEx`, `DirectoryInfoEx` and `FileInfoEx` classes.
  - Added `BeforeInvoke` event to `ContextMenuWrapper` class.
  - Added Run behavior when double click in filelist.
  - Added `FileSystemWatcherEx.Filter`.

- 01-20-10 Version 0.11
  - Added: `DirectoryTree` in the demo; now properly refreshes when changed. (Implemented an `ObservableCollection` in the `GetDirectoriesConverterEx` class using the `FileSystemWatcherEx` class.)
  - Added: `ContextMenuWrapper.OnQueryMenuItems.QueryContextMenu2 / QueryContextMenu3` property.
  - Added: `ContextMenuWrapper.OnBeforePopup` event.
  - Added: `ContextMenuWrapper.OnQueryMenuItems` event; now supports multilevel menu (e.g.: @"Tools\Add").
  - Added: `ContextMenuWrapper.OnQueryMenuItems` event; now supports GrayedItems / HiddenItems.
- 02-15-10 Version 0.12
  - Fixed: Fullname of User/Shared directory under desktop is now its GUID instead of its file path.
  - Fixed: PIDL, PIDLRel, ShellFolder, Storage properties generated on demand to avoid x-thread issues.
  - Added: `PathEx` class to deal with PIDL related paths.
- 03-14-10 Version 0.13
  - Fixed: `FileSystemWaterEx` ignoring remove directory event.
  - Fixed: Removed `IDisposable` in PIDL as it is causing an `AccessViolationException`, user has to free by calling the `Free()` method.
- 03-16-10 Version 0.14
  - Fixed: `FileSystemInfoEx` now stores a copy of PIDL/Rel, will return copy of it when properties are called (to avoid `AccessViolation`).
  - Fixed: `FileSystemInfoEx` records the PIDL when constructed, as some paths are not parseable (e.g., EntireNetwork).
  - Added: Allows folder list so Non-FileAncestor directories (e.g., recycle-bin) are listed.
- 03-18-10 Version 0.15
  - Fixed: ShellFolder/PIDL not freed in a couple locations.
  - (Please note that PIDL/ShellFolder/Store are no longer stored in the `FileSystemInfoEx` => must be freed by the user.)
- 03-19-10 Version 0.16
  - Added: `IShellFolder2` interface and `ShellFolder2` class.
  - Added: `ExtraPropertiesProvider` which can list the extra file properties / list columns available (e.g., `ExtraPropertiesProvider.GetProperty(file, ref ImageSummaryInformation.BitDepth);`).
  - Fixed: `getRelPIDL()` cannot return the correct value in the `File/DirInfoEx` construct with string. (Attempt to return a freed up pointer.)
  - Fixed: `ShellFolder` not freed in two spots.
- 04-26-2010 Version 0.17
  - Added `this` operator in `DirectoryInfoEx`.
  - Added `DefaultItem` and `DefaultCommand` in `BeforePopup`
  - Added `WorkSpawner`, which can spawn `ListWork`, `CopyWork`, `MoveWork`

  and `DeleteWork` to perform responsive threaded operations.
  - Fixed some XP system cannot create shared directories. (by cwharmon)
  - Removed `DirectoryInfoEx` file/directory `list caching` (_cachedFileList) as it slow down if too many files (and the old EnumDirs/EnumFiles implementation).
  - Added `DirectoryInfoEx/DirectoryEx.EnumerateFiles /EnumerateDirectories/EnumerateFileSystemInfos`() methods which work similar as the one in .Net4 (CancelDelegate is added to make it cancelable.)
  - Added FileEx.`ReadLines`/`ReadAllLines`() methods.
  - Added IOTools.`CopyFile`() method which support cancel.
  - Added `FileSystemInfoEx.FromString`() method.

- 05-25-2010 Version 0.18
  - WPF File Explorer User Control (`DirectoryTree` and `FileList`) now available.
  - Fixed `DirectoryInfoEx.EnumerateDirectories` return files when listing network directories.
  - Added `ExComparer` class, which enable sorting an array of `FileSystemInfoEx` entries.
  - Fixed `DriveInfoEx` return incorrect `TotalSize`. `DriveInfoEx` constructor now accept full drive name ("C" and "C:\" both accepted now)
  - (REMOVED) Added a check for NonEnumerated items so`DirectoryInfoEx.EnumerateDirectories` wont return some system directories (e.g. C:\MSOCache)
  - Added IOTools.`GetRelativePath`. Documented IOTools's static methods.
  - Added IOTools.`ShellFolderToCSIDL`() and `CSIDLToShellFolder`() static methods.
  - Added constructor of `DirectoryInfoEx` which support `Environment.ShellFolder`.
  - Fixed DirectoryInfoEx.`EnumerateFiles` ignore SearchPattern.
  - Fixed Context menu disappear in some case. (By cwharmon)
  - Updated `DirectoryInfoEx`/`FileInfoEx` listing code to improve speed.
  - Added Progress Dialog for all Work, WorkBase.`IsProgressDialogEnabled` and `ProgressDialog`.

- 07-17-2010 Version 0.19
  - Added `FileTypeInfoProvider`
  - Fixed `ArgumentException` when getting storage of C:\{User}\Desktop.

- 08-22-2010 Version 0.20
  - Fixed `ShellProgressDialog` still running after closed.
  - Added `LinkSummaryInformation` in `ExtraPropertiesProvider`.

- 11-04-2010 Version 0.21
  - Small update to `CustomMenuStructure` class.
  - Fixed FileSystemInfoEx.`getRelativePIDL`() and `getParentPIDL`() that return `relPIDL` that is not a clone, which will crash if attempted to Free. (e.g. in DirectoryEx.`Exists`())

- 03-01-2011 Version 0.22
  - Added `ImageExtractor`, which uses `IExtractImage` to generate thumbnails.
  - Added `PreviewHelper` and `PreviewControl`, which is a `IPreviewHandler`.
  - Fix illegal PIDL for Directory under Library.ms directory

- 12-16-2013 Version 0.24
  - Change access to FileSystemInfoEx.PIDL/PidlRel to extension methods (`RequestPIDL`/`RelPIDL`)
  - Fixed a memory leak when listing.
- 12-24-2013 Version 0.25
  - Fixed OpenWithInfo.`GetExecutablePath`()
- 11-10-2014 Version 1.0.26
  - Fixed two crashes related to x64 mode.
  - Added DirectoryInfoEx.ShellFolderType (`Environment.SpecialFolder`).
- 11-19-2014 Version 1.0.27
  - Added DirectoryInfoEx.`GetFileSystemInfosAsync`(), `GetDirectoriesAsync`() and `GetFilesAsync`() async methods.
  - Added KnownFolder support (See MSDN)

License

Share

About the Author

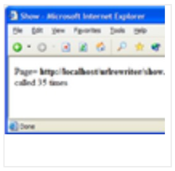
Leung Yat Chun

Software Developer

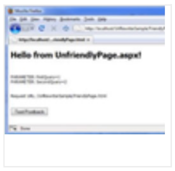
Hong Kong 🇭🇰

DirectoryInfoEx - [1.0.27]
WPF FileExplorer3 - [3.0.19]
WPF HtmlTextBlock - [Codeplex]
~~WPF ListView MultiSelect - [0.4]~~
WPF UIEventHub MultiSelect/DragDrop w Touch support - [3.0]
~~WPF BreadcrumbFolderTextBox - [2.5]~~
WPF BreadcrumbTree and Breadcrumb - [dirkster's edition]
WPF Aero Titlebar - [0.2]

You may also be interested in...


URL Rewriting with ASP.NET

Is SQL Server killing your application's performance?


URL Rewriting using ASP.NET for SEO

SAPrefs - Netscape-like Preferences Dialog

URL rewriting using ASP.NET routing

Window Tabs (WndTabs) Add-In for DevStudio

Comments and Discussions

50 messages have been posted for this article Visit **http://www.codeproject.com/Articles/39224/Rewrite-DirectoryInfo-using-IShellFolder** to post and view comments on this article, or click **here** to get a print view with messages.