|  |  |  |  |
| --- | --- | --- | --- |
|  | 11.01.2009 |  | |
|  | |  | |
| Использование Shell  *Используем Shell для получения списка файлов/папок* | | | |
|  |  | |  |
|  |  | | Михаил Фленов (http://www.flenov.info) |

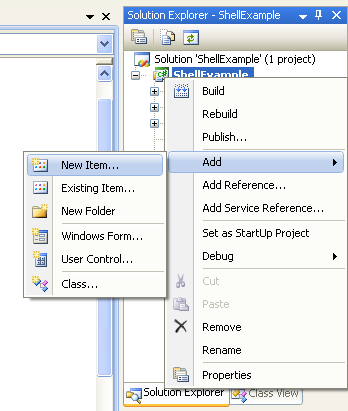
Использование Shell

Используем Shell для получения списка файлов/папок

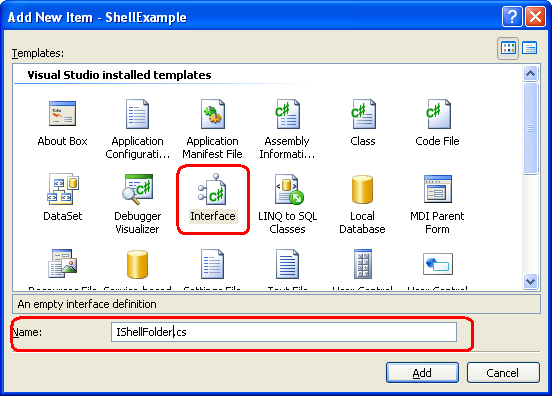
Прекрасный вопрос. Я рекомендую использовать системные функции. Это немного сложнее, потому что придется описывать необходимые системные интерфейсы, функции и структуры, которые в .NET не описаны. Но зато результат будет мощным и универсальным. В системе много функций работы с файловой системой, но какие из них нам понадобятся? Я помогу вам разобраться.

Если тебе нужно просто получить список файлов в директории, то лучше воспользоваться классом Directory. Совместно с File он предоставит тебе базовые возможности для работы с файловой системой. Но этого не всегда достаточно, и если нужно что-то большее - хорошие системные иконки, снимки файлов и т.д., то придётся опускаться на системный уровень, о чем мы и поговорим в этой статье.

Создайте новый проект и поместите на форму компонент ListView. В него мы будет загружать имена файлов и папок, которые найдем на рабочем столе. Теперь мы должны описать два интерфейса, которые нам понадобяться: IShellFolder и IEnumIDList. Создайте новый файл для интерфейса IShellFolder. Чтобы сделать это щелкните правой кнопкой по имени проекта в окне Solution Explorer и выберите меню New Item из меню Add. Перед вами откроется окно Add New:



Выберите иконку Interface в списке Templates и введите имя нового интерфейса IShellFolder в поле Name.



Сохраните новый файл под именем IShellFolder. Содержимое файла должно быть таким (можете скопировать его из браузера):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Runtime.InteropServices;

namespace ShellExample

{

[ComImport]

[InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]

[Guid("000214E6-0000-0000-C000-000000000046")]

public interface IShellFolder

{

[PreserveSig]

Int32 ParseDisplayName(

IntPtr hwnd,

IntPtr pbc,

[MarshalAs(UnmanagedType.LPWStr)]

string pszDisplayName,

ref uint pchEaten,

out IntPtr ppidl,

ref ShellAPI.SFGAO pdwAttributes);

[PreserveSig]

Int32 EnumObjects(

IntPtr hwnd,

ShellAPI.SHCONTF grfFlags,

out IntPtr enumIDList);

[PreserveSig]

Int32 BindToObject(

IntPtr pidl,

IntPtr pbc,

ref Guid riid,

out IntPtr ppv);

[PreserveSig]

Int32 BindToStorage(

IntPtr pidl,

IntPtr pbc,

ref Guid riid,

out IntPtr ppv);

[PreserveSig]

Int32 CompareIDs(

IntPtr lParam,

IntPtr pidl1,

IntPtr pidl2);

[PreserveSig]

Int32 CreateViewObject(

IntPtr hwndOwner,

Guid riid,

out IntPtr ppv);

[PreserveSig]

Int32 GetAttributesOf(

uint cidl,

[MarshalAs(UnmanagedType.LPArray)]

IntPtr[] apidl,

ref ShellAPI.SFGAO rgfInOut);

[PreserveSig]

Int32 GetUIObjectOf(

IntPtr hwndOwner,

uint cidl,

[MarshalAs(UnmanagedType.LPArray)]

IntPtr[] apidl,

ref Guid riid,

IntPtr rgfReserved,

out IntPtr ppv);

[PreserveSig()]

Int32 GetDisplayNameOf(

IntPtr pidl,

ShellAPI.SHGNO uFlags,

IntPtr lpName);

[PreserveSig]

Int32 SetNameOf(

IntPtr hwnd,

IntPtr pidl,

[MarshalAs(UnmanagedType.LPWStr)]

string pszName,

ShellAPI.SHGNO uFlags,

out IntPtr ppidlOut);

}

}

Теперь создайте еще один файл интерфейса с именем IEnumIDList. Его код должен быть таким:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Runtime.InteropServices;

namespace ShellExample

{

[ComImportAttribute()]

[InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]

[Guid("000214F2-0000-0000-C000-000000000046")]

public interface IEnumIDList

{

[PreserveSig()]

Int32 Next(

int celt,

out IntPtr rgelt,

out int pceltFetched);

[PreserveSig()]

Int32 Skip(

int celt);

[PreserveSig()]

Int32 Reset();

[PreserveSig()]

Int32 Clone(

out IEnumIDList ppenum);

}

}

Теперь у нас есть необходимые интерфейсы, но нужны еще функции и структуры. Давайте создадим файл ShellAPI.cs в котором и опишем необходимые функции. Чтобы сделать это, щелкните правой кнопкой мыши по имени проекта в окне Solution Explorer, как мы это делали для создания интерфейсов. Выберите New Item из меню Add. Снова появиться окно Add New Item. Теперь выберите иконку Class в списке Templates и напишите имя класса ShellAPI в поле Name. Код нового файла должен быть таким:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Runtime.InteropServices;

namespace ShellExample

{

public class ShellAPI

{

public const int S\_OK = 0;

public const int S\_FALSE = 1;

public const int MAX\_PATH = 260;

public static int cbFileInfo = Marshal.SizeOf(typeof(SHFILEINFO));

[Flags]

public enum SHCONTF

{

FOLDERS = 0x0020,

NONFOLDERS = 0x0040,

INCLUDEHIDDEN = 0x0080,

INIT\_ON\_FIRST\_NEXT = 0x0100,

NETPRINTERSRCH = 0x0200,

SHAREABLE = 0x0400,

STORAGE = 0x0800,

}

[Flags]

public enum SHGNO

{

NORMAL = 0x0000,

INFOLDER = 0x0001,

FOREDITING = 0x1000,

FORADDRESSBAR = 0x4000,

FORPARSING = 0x8000

}

[Flags]

public enum SHGFI : uint

{

ADDOVERLAYS = 0x20,

ATTR\_SPECIFIED = 0x20000,

ATTRIBUTES = 0x800,

DISPLAYNAME = 0x200,

EXETYPE = 0x2000,

ICON = 0x100,

ICONLOCATION = 0x1000,

LARGEICON = 0,

LINKOVERLAY = 0x8000,

OPENICON = 2,

OVERLAYINDEX = 0x40,

PIDL = 8,

SELECTED = 0x10000,

SHELLICONSIZE = 4,

SMALLICON = 1,

SYSICONINDEX = 0x4000,

TYPENAME = 0x400,

USEFILEATTRIBUTES = 0x10

}

[Flags]

public enum SFGAO : uint

{

BROWSABLE = 0x8000000,

CANCOPY = 1,

CANDELETE = 0x20,

CANLINK = 4,

CANMONIKER = 0x400000,

CANMOVE = 2,

CANRENAME = 0x10,

CAPABILITYMASK = 0x177,

COMPRESSED = 0x4000000,

CONTENTSMASK = 0x80000000,

DISPLAYATTRMASK = 0xfc000,

DROPTARGET = 0x100,

ENCRYPTED = 0x2000,

FILESYSANCESTOR = 0x10000000,

FILESYSTEM = 0x40000000,

FOLDER = 0x20000000,

GHOSTED = 0x8000,

HASPROPSHEET = 0x40,

HASSTORAGE = 0x400000,

HASSUBFOLDER = 0x80000000,

HIDDEN = 0x80000,

ISSLOW = 0x4000,

LINK = 0x10000,

NEWCONTENT = 0x200000,

NONENUMERATED = 0x100000,

READONLY = 0x40000,

REMOVABLE = 0x2000000,

SHARE = 0x20000,

STORAGE = 8,

STORAGEANCESTOR = 0x800000,

STORAGECAPMASK = 0x70c50008,

STREAM = 0x400000,

VALIDATE = 0x1000000

}

[Flags]

public enum FILE\_ATTRIBUTE

{

READONLY = 0x00000001,

HIDDEN = 0x00000002,

SYSTEM = 0x00000004,

DIRECTORY = 0x00000010,

ARCHIVE = 0x00000020,

DEVICE = 0x00000040,

NORMAL = 0x00000080,

TEMPORARY = 0x00000100,

SPARSE\_FILE = 0x00000200,

REPARSE\_POINT = 0x00000400,

COMPRESSED = 0x00000800,

OFFLINE = 0x00001000,

NOT\_CONTENT\_INDEXED = 0x00002000,

ENCRYPTED = 0x00004000

}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]

public struct SHFILEINFO

{

public IntPtr hIcon;

public int iIcon;

public SFGAO dwAttributes;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = MAX\_PATH)]

public string szDisplayName;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 80)]

public string szTypeName;

}

[DllImport("shell32.dll")]

public static extern Int32 SHGetDesktopFolder(

out IntPtr ppshf

);

[DllImport("shell32", EntryPoint = "SHGetFileInfo",

ExactSpelling = false, CharSet = CharSet.Auto,

SetLastError = true)]

public static extern IntPtr SHGetFileInfo(

string pszPath,

FILE\_ATTRIBUTE dwFileAttributes,

ref SHFILEINFO sfi,

int cbFileInfo,

SHGFI uFlags);

[DllImport("shell32", EntryPoint = "SHGetFileInfo",

ExactSpelling = false, CharSet = CharSet.Auto,

SetLastError = true)]

public static extern IntPtr SHGetFileInfo(

IntPtr ppidl,

FILE\_ATTRIBUTE dwFileAttributes,

ref SHFILEINFO sfi,

int cbFileInfo,

SHGFI uFlags);

}

}

Теперь у нас есть все необходимое и можно переходить к рассмотрению функций получения списка файлов. Комментарии помогут вам разобраться с кодом:.

// получить папку рабочего стола

IntPtr shellFolderPtr;

ShellAPI.SHGetDesktopFolder(out shellFolderPtr);

IShellFolder shellFolder =

(IShellFolder)Marshal.GetTypedObjectForIUnknown(shellFolderPtr, typeof(IShellFolder));

// очистить содержимое listView1

listView1.Items.Clear();

// мы будем перечислять папки и файлы

ShellAPI.SHCONTF fileFlag = ShellAPI.SHCONTF.NONFOLDERS |

ShellAPI.SHCONTF.FOLDERS;

// если результат равен S\_OK, то все прекрасно

IntPtr fileEnumPtr = IntPtr.Zero;

if (shellFolder.EnumObjects(IntPtr.Zero, fileFlag, out fileEnumPtr) == ShellAPI.S\_OK)

{

IEnumIDList fileEnum = (IEnumIDList)Marshal.GetTypedObjectForIUnknown(

fileEnumPtr, typeof(IEnumIDList));

IntPtr gelt;

int celtFetched;

// перечисляем найденные объекты

while (fileEnum.Next(1, out gelt, out celtFetched) == ShellAPI.S\_OK

&& celtFetched == 1)

{

// получить имя текущего объекта

ShellAPI.SHFILEINFO info = new ShellAPI.SHFILEINFO();

ShellAPI.SHGetFileInfo(gelt, 0, ref info, ShellAPI.cbFileInfo,

ShellAPI.SHGFI.DISPLAYNAME | ShellAPI.SHGFI.TYPENAME);

// добавить его в список

listView1.Items.Add(info.szDisplayName);

}

}

