

Simple shell context menu

Simple shell context menu

Demonstrates how to create a simple shell context menu using a few registry entries, instead of COM. The sample context menu creates a grayscale copy of the selected JPEG image.

Append items to Explorer Shell context menu easily – How to add ?

Append an entry to Windows Explorer Shell context menu easily with Windows Explorer Shell Context Menu. This powerful .Net component for custom items adding to Explorer Shell context menu will add all your custom application items to Windows Explorer Shell context menu. This .Net component with full C# and Visual Basic .NET support include detailed C# and VB.NET samples, tutorials , user-friendly manuals and support all you may need :

- Add items to Windows Explorer Shell context menu to be shown on any Windows OS (all operating systems are supported – Windows XP, Vista, Windows x64 of all types , etc.)
- Add items to Windows Explorer Shell context menu to be shown in any way - with your custom caption and your custom icon, as separator or sub-menu
- Add your items to Windows Explorer Shell context menu to be shown for all files or shown only for files of particular type (for example, only for .DOC , .MP3,.WMA,.AAC , .WMV files)
- Add your program items to Windows Explorer Shell context menu, sub-menus, sub-sub-menus, sub-menus of unlimited depth and even more

Windows Explorer Shell Context Menu - is a powerful .Net framework component that support all you need to insert all your items to Windows Explorer Shell context menu - in a fast and easy way. Add your application entries to Windows Explorer Shell context menu right now – add entries to context menu fast and exactly as you prefer :

Introduction

Lets discuss, how to add custom items to Windows Explorer Shell context menus. Shell context menus are displayed when you right click on shell objects such as files and folders. A full-blown context menu is a COM object that implements the IContextMenu and IShellExtInt interfaces. Because you most likely develop applications and want to add custom items not to outdated operating systems, please note that this method works only for Windows 95 / Windows 98 (not on XP, Vista, x64 - 64-bit Windows), to add items to Windows Explorer Shell context menu you should use, according to Microsoft guidelines, appropriate .Net component - Windows Explorer Shell Context Menu. This article demonstrates how to create a simple shell context menu (also called a shortcut menu) that does not require COM and only requires a few registry entries.

Registry entries

You can hookup a context menu to any file type by adding entries under the HKEY_CLASSES_ROOT\

REGEDIT4

```
[HKEY_CLASSES_ROOT\dllfile\shell]
[HKEY_CLASSES_ROOT\dllfile\shell\Register]
[HKEY_CLASSES_ROOT\dllfile\shell\Register\command]
@="regsvr32 \"%L\""
```

```
[HKEY_CLASSES_ROOT\dllfile\shell\Unregister]
[HKEY_CLASSES_ROOT\dllfile\shell\Unregister\command]
```

```
@="regsvr32 /u \"%L\""
```

A view of the registry is shown below. The HKEY_CLASSES_ROOT\dlfile\shell key contains the list of context menus for DLL files. The Register and Unregister keys are two of the menus for DLL files (these also specify the menu text since a default value is not specified). The default value of the command key specifies the command line that is executed when the context menu is invoked. The %L argument is a placeholder to the full path of the selected item. You can read more about the registry settings at the MSDN article: [Extending Shortcut Menus](#).

Registering and un-registering

The sample application contains the FileShellExtension class that registers and un-registers a simple shell context menu. The Register method creates the necessary registry entries, and the Unregister method removes the registry entries.

```
static class FileShellExtension
{
    public static void Register(string fileType,
        string shellKeyName, string menuText, string menuCommand)
    {
        // create path to registry location

        string regPath = string.Format(@"{0}\shell\{1}",
            fileType, shellKeyName);

        // add context menu to the registry

        using (RegistryKey key =
            Registry.ClassesRoot.CreateSubKey(regPath))
        {
            key.SetValue(null, menuText);
        }

        // add command that is invoked to the registry

        using (RegistryKey key = Registry.ClassesRoot.CreateSubKey(
            string.Format(@"{0}\command", regPath)))
        {
            key.SetValue(null, menuCommand);
        }
    }

    public static void Unregister(string fileType, string shellKeyName)
    {
        Debug.Assert(!string.IsNullOrEmpty(fileType) &&
            !string.IsNullOrEmpty(shellKeyName));

        // path to the registry location

        string regPath = string.Format(@"{0}\shell\{1}",
            fileType, shellKeyName);

        // remove context menu from the registry

        Registry.ClassesRoot.DeleteSubKeyTree(regPath);
    }
}
```

The sample application self-registers when executed without any command line arguments, or with the -register command; it unregisters when the -unregister command is specified. The usage of the FileShellExtension class is shown below.

```
// sample usage to register
```

```
// get full path to self, %L is a placeholder for the selected file

string menuCommand = string.Format("{0} \\\"%L\"",
    Application.ExecutablePath);
FileShellExtension.Register("jpegfile", "Simple Context Menu",
    "Copy to Grayscale", menuCommand);

// sample usage to unregister

FileShellExtension.Unregister("jpegfile", "Simple Context Menu");
```

Creating a grayscale image

The CopyGrayscaleImage method is called when the context menu is clicked. The ColorMatrix class is used to generate a grayscale copy of the selected image.

Collapse

```
static void CopyGrayscaleImage(string filePath)
{
    // full path to the grayscale copy

    string grayFilePath = Path.Combine(
        Path.GetDirectoryName(filePath),
        string.Format("{0} (grayscale){1}",
            Path.GetFileNameWithoutExtension(filePath),
            Path.GetExtension(filePath)));

    // using calls Dispose on the objects, important

    // so the file is not locked when the app terminates

    using (Image image = new Bitmap(filePath))
    using (Bitmap grayImage = new Bitmap(image.Width, image.Height))
    using (Graphics g = Graphics.FromImage(grayImage))
    {
        // setup grayscale matrix

        ImageAttributes attr = new ImageAttributes();
        attr.SetColorMatrix(new ColorMatrix(new float[][]{
            new float[]{0.3086F,0.3086F,0.3086F,0,0},
            new float[]{0.6094F,0.6094F,0.6094F,0,0},
            new float[]{0.082F,0.082F,0.082F,0,0},
            new float[]{0,0,0,1,0,0},
            new float[]{0,0,0,0,1,0},
            new float[]{0,0,0,0,0,1}}));

        // create the grayscale image

        g.DrawImage(image, new Rectangle(0, 0, image.Width, image.Height),
            0, 0, image.Width, image.Height, GraphicsUnit.Pixel, attr);

        // save to the file system

        grayImage.Save(grayFilePath, ImageFormat.Jpeg);
    }
}
```

The original and generated grayscale images are shown below:

Running the sample

The sample was built with Visual Studio 2005, and requires the .NET Framework 2.0; however, the ideas can easily be incorporated into any .NET version and language. You can do the following to run the sample:

- * Build and run the application. This registers the context menu by adding the HKEY_CLASSES_ROOT\jpegfile\shell\Simple Context Menu key to the registry.
- * Right click on a JPEG file, you should see a new Copy to Grayscale context menu.
- * Click the context menu to create a grayscale copy of the image.
- * Unregister the context menu by running SimpleContextMenu.exe –unregister.

Context menus for all files, folders, and drives

You can also hookup context menus to all files, folders, and drives by adding entries to the file type's *, Directory, and Drive registry keys. For example, XP PowerToys adds the Open Command Window Here menu to all folders with the following registry script:

REGEDIT4

```
[HKEY_CLASSES_ROOT\Directory\shell\cmd]
@="Open Command Window Here"
```

```
[HKEY_CLASSES_ROOT\Directory\shell\cmd\command]
@="cmd.exe /k \"%cd %L\""
```

Simple context menus do have limitations, such as only accepting one file on the command line, but they are still pretty useful. I'll cover creating a full-blown .NET COM-based context menu in a future article.

License

This article has no explicit license attached to it but may contain usage terms in the article text or the download files themselves. If in doubt please contact the author via the discussion board below.

A list of licenses authors might use can be found [here](#)