

Creating Explorer Context Menu using C#

Creating and Installing Windows Explorer Shell Context Menu using C#

Introduction

This article basically illustrates how to create an Windows Explorer Shell context menu item upon installation using C#. Usually such projects are not standalone, and are included upon other features, but this will show a step by step guide to create one.

Add an item to Windows Explorer context (right-click) menu easily – How ?

Add items to Windows Explorer context menu easily with Windows Explorer Shell Context Menu. This powerful component for custom items adding to Windows Explorer Shell context menu will add all your custom application entries to Explorer context menu. It , C++ and VB.NET support include detailed C# / 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 OS (all operating systems are supported – Windows XP, Vista, Windows x64 of all types , etc.)
- Add any type of items to Windows Explorer Shell context menu to be shown in any way - with custom caption and your custom icon, as separator or sub-menu
- Add items to Explorer Shell context menu to be shown for all files or shown only for computer files of particular type (for example, only for .PDF .TXT , .MP3,.WMA,.AAC , .MPG media files)
- Add your program entries to Windows Explorer Shell context menu, sub-menus, sub-menus of unlimited depth and even much more

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

Background

To use this short tutorial you need to understand how to manually add an Windows Explorer Shell context menu item, and this is an interesting link to check.

Also you should note that this method works only for Windows 95 / Windows 98 (not on XP, Vista, x64 - 64-bit Windows), so if you are a .Net developer (C#, VB.NET, etc.) and you are searching for a simple way to add entries to Windows Explorer Shell context menu, then 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 developer-friendly .Net component will add entries of all types (including separators, sub-menus, etc.) to Windows Explorer Shell context menu in a very easy way.

Creating the Project

Your ingredients for such a project are:

1. C# Class Library
2. Setup Project

For the C# class library, delete the default class, right click on the project name and click Add New Item and choose installer class.

For the setup project, just add the primary output of the installer class project and add this output to all four default

custom actions. (Right click on the setup project name, select View > Custom action.)

At the end, your solution should look like we have.

Quite simplistic. Now we just need to adjust one more thing in the setup project before tackling the code. Basically we need to grab the target directory submitted by the user upon installation, to do this.

Select the Primary output added under the Install Action, and type in the CustomActionData.

Creating the Installer Class

The main purpose of the installer is to create the registry keys, so here goes:

Collapse

```
//
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration.Install;
//DON'T FORGET TO ADD THIS ONE - used for the Registry Class
using Microsoft.Win32;

namespace WindowsExplorerContextMenuContextInstallerClass
{
    [RunInstaller(true)]
    public partial class WindowsExplorerContextMenuInstaller : Installer
    {
        public WindowsExplorerContextMenuInstaller()
        {
            InitializeComponent();
        }

        public override void Install(System.Collections.IDictionary stateSaver)
        {
            base.Install(stateSaver);

            //grab the target director and add to install.installstate
            //note that the grabbed parameters are exactly the same
            //as the one we set above
            stateSaver.Add("TargetDir", Context.Parameters["DP_TargetDir"].ToString());
        }

        public override void Commit(System.Collections.IDictionary savedState)
        {
            //retrieve the saved state
            base.Commit(savedState);
            //adding items to the context menu of WindowsExplorerContextMenu
            RegistryKey key;
            //1,2,4,8,10,20 in heximal are 1,2,4,8,16,32 in decimal (respectively)
            string keyValueInt = "16";
            //the location of our key
            string subKey =
"@SOFTWARE\Microsoft\Windows Explorer\MenuExt\Sample Action";
            //create it
            key = Registry.CurrentUser.CreateSubKey(subKey);
            //set the value
            key.SetValue("Contexts", Convert.ToInt32(keyValueInt),
                RegistryValueKind.DWord);
            //set the path to the action file
            key.SetValue(null, "file://" + savedState["TargetDir"].ToString() +
                "\\action.html");
            //and close
            key.Close();
        }
    }
}
```

```

protected override void OnBeforeUninstall
    (System.Collections.IDictionary savedState)
    {
        base.OnBeforeUninstall(savedState);
        //removing items from the context menu of WindowsExplorerContextMenu
        string subKey =
        @"SOFTWARE\Microsoft\Windows Explorer\MenuExt\Sample Action";
        Registry.CurrentUser.DeleteSubKey(subKey);
    }
}
}

```

Now basically the installer class does the following:

1. Upon installation, grabs the target directory and saves it so we can use it at a later stage
2. Upon committing, creates the registry keys
3. In case the user wishes to uninstall, simply deletes the keys

Creating the Action File

This action file is the actual code we are running when clicking on the context menu:

```

<SCRIPT LANGUAGE = "JavaScript">
// Get the window object where the context menu was opened.
var oWindow = window.external.menuArguments;
// Get the document object exposed through oWindow.
var oDocument = oWindow.document;
// Get the selection from oDocument.
// in oDocument.
var oSelect = oDocument.selection;
// Create a TextRange from oSelect.
var oSelectRange = oSelect.createRange();
// Get the text of the selection.
var sNewText = oSelectRange.text;
// If something is selected, alert showing the selected text.
if(sNewText.length != 0){
    alert(sNewText);
}
</SCRIPT>

```

The above is a simple JavaScript that grabs the selected text and show it in a message box... you can have other actions depending on your needs. (The code above is based on this.)

Installing

Once you have completed all of the above, build your solution and install.

You should have the following key added to your registry.

If you open a new instance of Windows Explorer and right click on a selected text, you should be getting.

Recap

So basically you should now be able to create and delete registry keys upon installation with no problem at all.

Note

Now this is definitely not something innovative or out of the ordinary, but I couldn't find something similar on The ContextMenu.net, so I thought of adding it.

Additionally, this is a compilation of items I found while "Googling" here and there...