*Design Windows Explorer Extensions*

|  |  |
| --- | --- |
| Bert O’Neill | |
| Department: | Software Demo’s |
| Document Title: | Design Windows Explorer Extensions with SharpShell |
| Author: | Bert O’Neill |
| Date: | 21-Jan-2023 |
| Version | 1.0 |
| SharePoint |  |
| Document Goals | Provide a detailed overview for developers who wish to extend the context menu of their Windows Explorer shell. |

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 21-Jan-2023 | 1.0 | Initial draft | Bert O’Neill |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Revision History 1](#_Toc125375069)

[Introduction 3](#_Toc125375070)

[Purpose 3](#_Toc125375071)

[Scope 3](#_Toc125375072)

[Prerequisites 3](#_Toc125375073)

[Solution and Environment Setup 4](#_Toc125375074)

[Codebase 4](#_Toc125375075)

[Shell ExViewer 4](#_Toc125375076)

[Design Custom Windows Explorer ContextMenu Extension 5](#_Toc125375077)

[Add ShapShell Nuget Package 7](#_Toc125375078)

[Strongly Sign Your Project 8](#_Toc125375079)

[Compile Project for Release 9](#_Toc125375080)

[Registering and Unregistering Assembly 9](#_Toc125375081)

[Register 9](#_Toc125375082)

[Verify Assembly Registered 10](#_Toc125375083)

[Unregister 11](#_Toc125375084)

[Testing and Debugging Your Windows Explorer Extension 12](#_Toc125375085)

[Useful Links 12](#_Toc125375086)

# Introduction

This document provides a detailed overview on how to implement Windows Explorer extensions\Add-ins into your Windows environment. Touching on the development and installation practices.

## Purpose

This document provides a comprehensive technical overview on how to integrate your own custom extension into your own Windows environment – what components are involved and how they are related to each other.

## Scope

The scope of this document is to convey the concepts needed to produce a Windows extension using the SharpShell assembly.

# Prerequisites

* A good understanding in OOD (especially inheritance)
* A good understanding of the .Net Framework (4.0+)
* Knowledge on how to register assemblies
* Knowledge on using Visual Studio

# Solution and Environment Setup

## Codebase

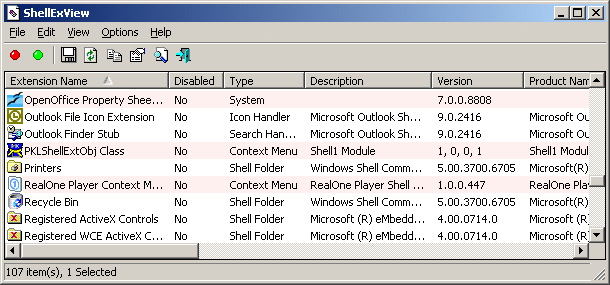
Clone the code-repo using this link <https://github.com/Bert0Neill/Extensions.git>.

## Shell ExViewer

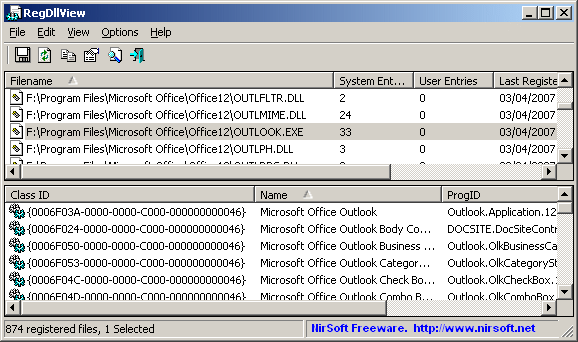
Navigate to the following [location](https://www.nirsoft.net/utils/shexview.html), download and unzip the ShellExView application. Once unzipped, you simply run the executable file *shexview.exe.* This will load all your registered assemblies (with your local environment).

<https://www.nirsoft.net/utils/shexview.html>

Later, we will use this application (after refreshing it) to find you newly created assembly, and verify that it got registered correctly.



There’s also another tool (like ShellExViewer) that you can use, but I like sing the ShellExViewer. It can be downloaded from here <https://www.nirsoft.net/utils/registered_dll_view.html> both tools do the same job – verifying that your assembly got registered.

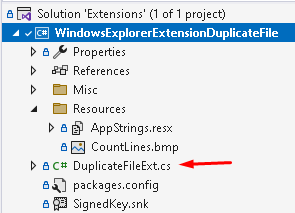


# Design Custom Windows Explorer ContextMenu Extension

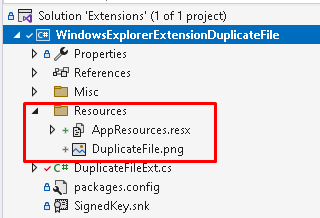
Load the project up in Visual Studio and you can use this as a guide when developing your own extension.

Create an empty Solution and add a new Class Library project to it. Rename the default Class.cs file to the feature of your new extension.

In my case, I am going to create an extension that will duplicate a file, by using Windows Explorer’s context menu.



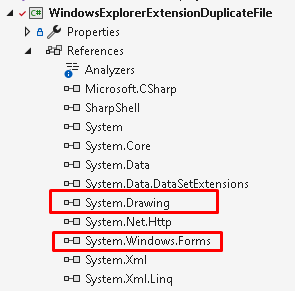
I’ve added a resource file to hold any string and Icons that I wish to render in the UI:



Add the following references to the Class project:

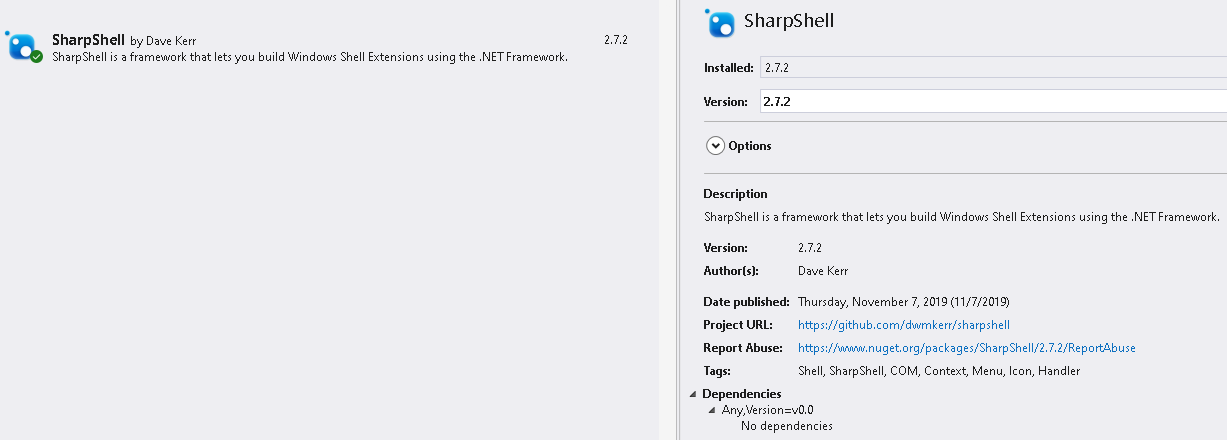
1. System.Windows.Forms
2. System.Drawing

System.Windows.Forms is needed because we're going to use the WinForms ContextMenuStrip to define the context menu. System.Drawing is needed as we're going to want to use Icons.



## Add ShapShell Nuget Package

Add the SharpShell Nuget package to your Class library project, this assembly does a lot of the plumbing for us, and you can create much more than just ContextMenu’s using this package.

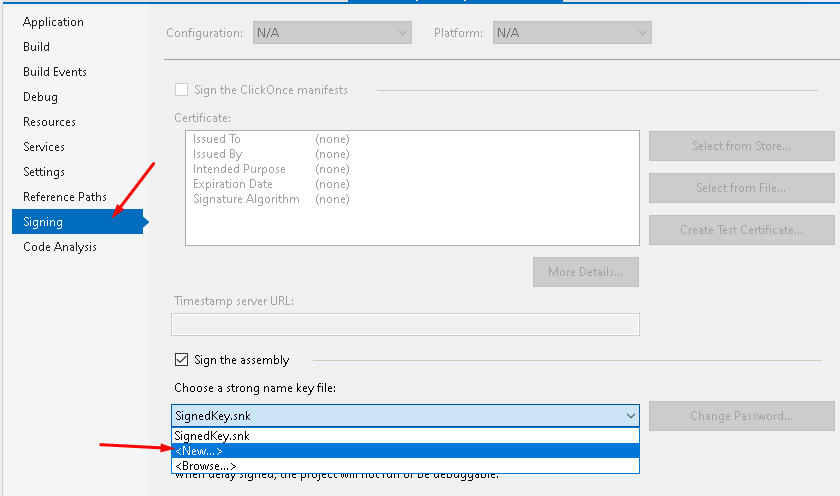


Take a look at the SharpShell Git Hub site, and the specific ContextMenu link to see what else your extensions can do.

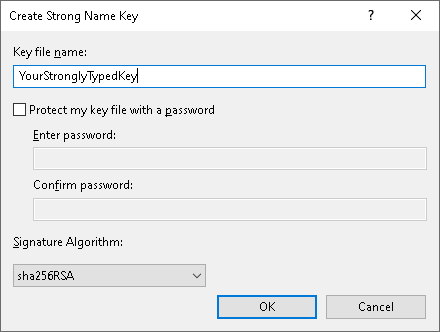
* <https://github.com/dwmkerr/sharpshell>
* <https://github.com/dwmkerr/sharpshell/blob/main/docs/context-menu.md>
* <https://github.com/dwmkerr/sharpshell/blob/main/SharpShell/SharpShell/SharpContextMenu/SharpContextMenu.cs>

# Strongly Sign Your Project

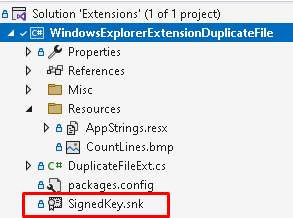
Bring up the properties pane of your project, select *Signing* and then select New from the *Choose String Name* dropdown.



Enter in a key name (this can be anything) and in this example I have not chosen to use a password, but if you are registering your extension within your network domain, it should be password protected.

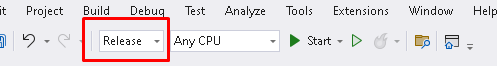


Below you can see that I called my key name *SignedKey*.



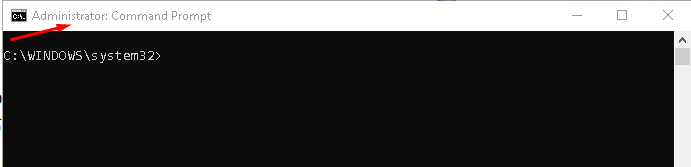
# Compile Project for Release

When you are happy with your code and you have the strongly typed key added, you are ready to compile your extension – do this in release mode.



# Registering and Unregistering Assembly

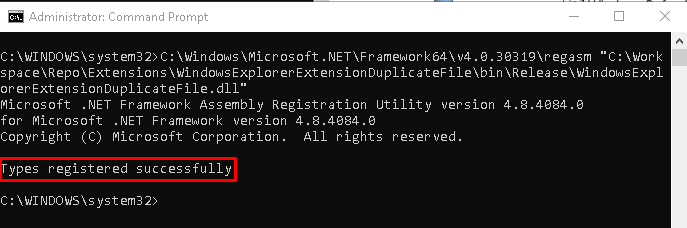
Open a Command Prompt in Admin Mode:



## Register

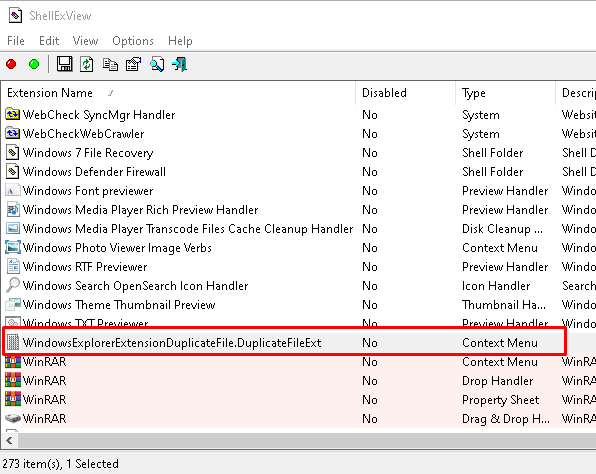
Run the following command in the command prompt (making sure to alter the path to your DLL). Also, note that the RegAsm file maybe located in a different folder depending on the version of .Net Framework you are using.

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\regasm "C:\Workspace\Repo\Extensions\WindowsExplorerExtensionDuplicateFile\bin\Release\WindowsExplorerExtensionDuplicateFile.dll"



## Verify Assembly Registered

Open the ShellExView application (refresh if already open), and search for the newly added extension WindowsExplorerExtensionDuplicateFile. If it is not present, then it didn’t register correctly.



**NB:** You will notice that some rows are coloured red, that is because they are not Microsoft assemblies (so your custom assembly will also be in red).

If the assembly didn’t register, then run the following command in the Command prompt with the following switch /codebase

"C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe" "C:\Workspace\Repo\Extensions\WindowsExplorerExtensionDuplicateFile\bin\Release\WindowsExplorerExtensionDuplicateFile.dll" /codebase

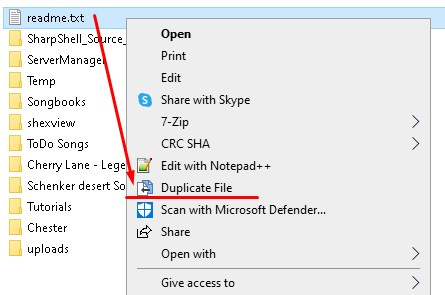
Refresh the ShellExViewer application and verify that it installed (the difference is that the file will not be added to the GAC, but a reference from the GAC to the assembly folder – thus making it more difficult to maintain, unless you are using a server location to hold all your extensions).

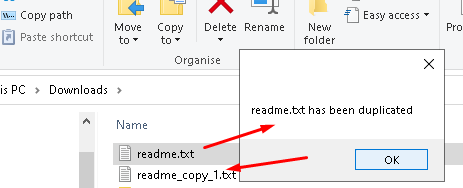
## Unregister

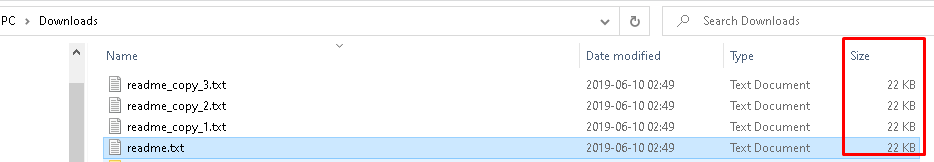
To unregister your assembly simply run the RegAsm again with the Uninstall switch at the end.

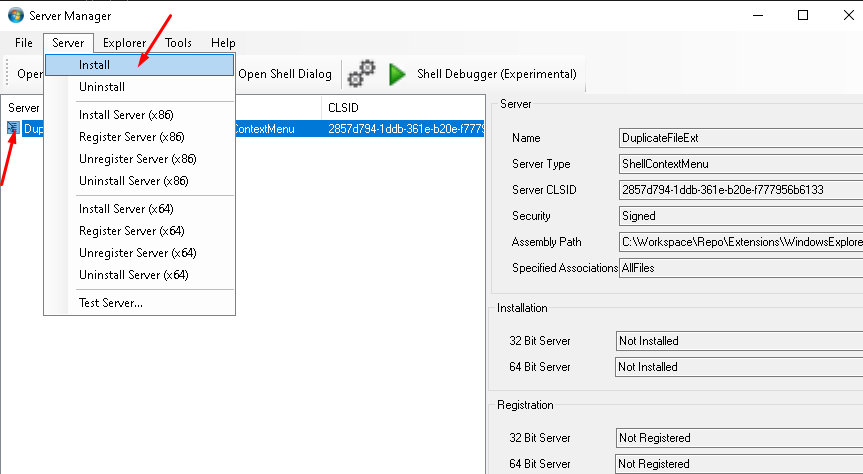
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\regasm "C:\Workspace\Repo\Extensions\WindowsExplorerExtensionDuplicateFile\bin\Release\WindowsExplorerExtensionDuplicateFile.dll" /U

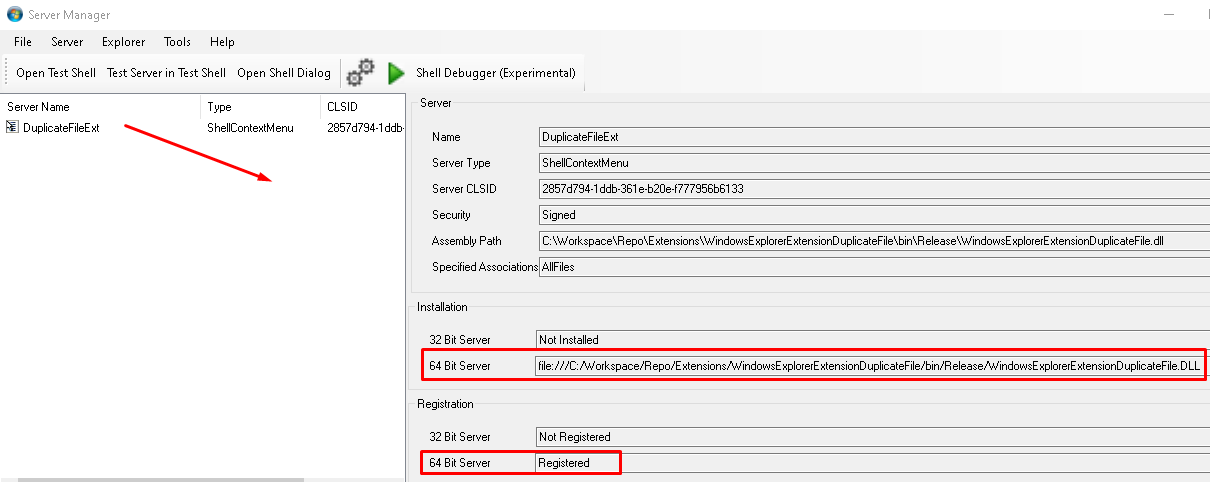
# Testing and Debugging Your Windows Explorer Extension











# Useful Links