Get file version and assembly version of DLL files in the current directory and all sub directories

I would like to be able to get the file version and assembly version of all DLL files within a directory and all of its subdirectories. I'm new to programming, and I can't figure out how to make this loop work.

I have this PowerShell code to get the assembly version (taken from a forum):

```
$strPath = 'c:\ADMLibrary.dll'
$Assembly = [Reflection.Assembly]::Loadfile($strPath)

$AssemblyName = $Assembly.GetName()
$Assemblyversion = $AssemblyName.version
```

And this as well:

```
$file = Get-ChildItem -recurse | %{ $_.VersionInfo }
```

How can I make a loop out of this so that I can return the assembly version of all files within a directory?

powershell

edited Dec 10 '14 at 19:43
Peter Mortensen **11.1k** 15 76 109

asked Jul 16 '10 at 16:30
Tim Sloyan **126** 1 2 3

2 Answers

Here is a pretty one liner:

```
Get-ChildItem -Filter *.dll -Recurse | Select-Object -ExpandProperty VersionInfo
```

In short for PowerShell version 2:

```
ls -fi *.dll -r | % { $_.versioninfo }
```

In short for PowerShell version 3 as suggested by tamasf:

```
ls *.dll -r | % versioninfo
```

edited Jan 9 '13 at 13:44

answered Sep 30 '12 at 18:45
knut **3,136** 2 24 40

6 In short: ls *.dll -r | % versioninfo – tamasf Jan 9 '13 at 8:56

3 This will only yield the FileVersion but not the AssemblyVersion – bitbonk Feb 5 '15 at 11:44

For managed dll's, to get the real/updated version info, .NET is really the only way to go.

elegantcode.com/2009/12/22/… Look at the "update" down below, the rest is over-complicating it :p – Christopher Wirt Jul 16 '15 at 17:24

As an ugly one-liner:

```
Get-ChildItem -Filter *.dll -Recurse |
    ForEach-Object {
        try {
            $_ | Add-Member NoteProperty FileVersion ($_.VersionInfo.FileVersion)
            $_ | Add-Member NoteProperty AssemblyVersion (
                [Reflection.AssemblyName]::GetAssemblyName($_.FullName).Version
            )
        } catch {}
        $_
    } |
    Select-Object Name,FileVersion,AssemblyVersion
```

If you only want the current directory, then obviously leave out the `-Recurse` parameter. If you want all files instead of just DLLs, then remove the `-Filter` parameter and its argument. The code is (hopefully) pretty straightforward.

I'd suggest you spin off the nasty parts within the `try` block into separate functions since that will make error handling less awkward here.

Sample output:

```
Name                                    FileVersion     AssemblyVersion
----                                    -----------     ---------------
Properties.Resources.Designer.cs.dll    0.0.0.0         0.0.0.0
My Project.Resources.Designer.vb.dll    0.0.0.0         0.0.0.0
WindowsFormsControlLibrary1.dll         1.0.0.0         1.0.0.0
WindowsFormsControlLibrary1.dll         1.0.0.0         1.0.0.0
WindowsFormsControlLibrary1.dll         1.0.0.0         1.0.0.0
```

edited Nov 4 '14 at 11:15

answered Jul 17 '10 at 1:05
Joey
221k 47 483 547

This works, but unfortunately it loads the files and never releases them ... Run it and then try to delete one of those files. – Karel Frajták Oct 15 '14 at 15:23

1 That is indeed a drawback. You can circumvent this by spawning a new PowerShell instance: `powershell -NoProfile -OutputFormat XML -EncodedCommand $encodedCommand |%{$_}` with `$encodedCommand` being the Base64-encoded variant of the above snippet (see `powershell /?` for a sample how to obtain it). This will yield the same objects that would otherwise be produced, but the shell loading the files is no longer alive now. – Joey Oct 16 '14 at 9:14

I am trying to load the assemblies into new AppDomain and unload it later, but it does not work - the assembly loader looks for assemblies in PowerShell directory (C:/windows/system32/WindowsPowerShell/v1.0) even though I set the base directory when creating the new domain... – Karel Frajták Oct 17 '14 at 12:13

1 I didn't say that! I just tried it and it locked those files. And then tried different approach and that didn't work. Yes, spawning new process is the solution here. – Karel Frajták Oct 17 '14 at 21:04

3 You can use `AssemblyName.GetAssemblyName` to get the AssemblyName (and hence the AssemblyVersion) without loading the assembly – lesscode Nov 3 '14 at 22:49