How to delete all files and folders in a directory?

Ask Question

566

95

Using C#, how can I delete all files and folders from a directory, but still keep the root directory?

c# .net

edited Sep 27 '13 at 17:09

Carlos Muñoz

12.8k 7 46 72

asked Aug 17 '09 at 15:48

JL.

31.9k 106 280 424

7 What would be nice if DirectoryInfo had a method like .Clean(); – JL. Aug 17 '09 at 15:54

5 or .DeleteFolders, and DeleteFiles methods. – JL. Aug 17 '09 at 15:54

17 You want to be aware that your Deletes could very easily throw an exception if a file is locked (or if you don't have rights). See the FileInfo.Delete for a list of the exceptions. – Shane Courtrille Aug 17 '09 at 15:56

29 Answers

712

```
System.IO.DirectoryInfo di = new DirectoryInfo("YourPath");

foreach (FileInfo file in di.GetFiles())
{
    file.Delete();
}
foreach (DirectoryInfo dir in di.GetDirectories())
{
    dir.Delete(true);
}
```

If your directory may have many files, `EnumerateFiles()` is more efficient than `GetFiles()`, because when you use `EnumerateFiles()` you can start enumerating it before the whole collection is returned, as opposed to `GetFiles()` where you need to load the entire collection in memory before begin to enumerate it. See this quote here:

> Therefore, when you are working with many files and directories, EnumerateFiles() can be more efficient.

The same applies to `EnumerateDirectories()` and `GetDirectories()`. So the code would be:

```
foreach (FileInfo file in di.EnumerateFiles())
{
    file.Delete();
}
foreach (DirectoryInfo dir in di.EnumerateDirectories())
{
    dir.Delete(true);
}
```

For the purpose of this question, there is really no reason to use `GetFiles()` and `GetDirectories()`.

edited May 29 '18 at 6:49

Andrew Morton

16.1k 6 35 53

answered Aug 17 '09 at 15:52

gsharp

15.1k 17 61 101

6 What's is about stackoverflow.com/questions/12415105/… "When you call Directory.Delete and a file is open in such way, Directory.Delete succeeds in deleting all files but when Directory.Delete calls RemoveDirectory a "directory is not empty" exception is thrown because there is a file marked for deletion but not actually deleted." – Kiquenet Jul 19 '13 at 7:54

64 DirectoryInfo is slow as this gathers much more other data. BTW: `Directory.Delete(path, true)` will take care of all :) – AcidJunkie Apr 1 '14 at 16:44

40 @AcidJunkie, That will also remove the directory in question, whereas the OP specifically asks for the root directory to be kept. – Marc L. May 30 '14 at 17:40

4 Note that this won't work if any of the files are read-only. You need to remove the read-only flag before calling `file.Delete()` . – Ben Apr 17 '15 at 8:54

7 This seems to not work if sub-directories contain files. – cdiggins Aug 4 '16 at 19:47

4

Here is the tool I ended with after reading all posts. It does

- Deletes all that can be deleted
- Returns false if some files remain in folder

It deals with

- Readonly files
- Deletion delay

Home

PUBLIC

- Locked files

It doesn't use Directory.Delete because the process is aborted on exception.

```
/// <summary>
/// Attempt to empty the folder. Return false if it fails (locke
/// </summary>
/// <param name="pathName"></param>
/// <returns>true on success</returns>
public static bool EmptyFolder(string pathName)
{
    bool errors = false;
    DirectoryInfo dir = new DirectoryInfo(pathName);

    foreach (FileInfo fi in dir.EnumerateFiles())
    {
        try
        {
            fi.IsReadOnly = false;
            fi.Delete();

            //Wait for the item to disapear (avoid 'dir not empt
            while (fi.Exists)
            {
                System.Threading.Thread.Sleep(10);
                fi.Refresh();
            }
        }
        catch (IOException e)
        {
            Debug.WriteLine(e.Message);
            errors = true;
        }
    }

    foreach (DirectoryInfo di in dir.EnumerateDirectories())
    {
        try
        {
            EmptyFolder(di.FullName);
            di.Delete();

            //Wait for the item to disapear (avoid 'dir not empt
            while (di.Exists)
            {
```

```
                    System.Threading.Thread.Sleep(10);
                    di.Refresh();
                }
            }
            catch (IOException e)
            {
                Debug.WriteLine(e.Message);
                errors = true;
            }
        }

        return !errors;
    }
```

answered May 18 '18 at 6:30

Eric Bole-Feysot

7,319 3 37 40

25

Based on the hiteshbiblog, you probably should make sure the file is read-write.

```
private void ClearFolder(string FolderName)
{
    DirectoryInfo dir = new DirectoryInfo(FolderName);

    foreach (FileInfo fi in dir.GetFiles())
    {
        fi.IsReadOnly = false;
        fi.Delete();
    }

    foreach (DirectoryInfo di in dir.GetDirectories())
    {
        ClearFolder(di.FullName);
        di.Delete();
    }
}
```

If you know there are no sub-folders, something like this may be the easiest:

```
Directory.GetFiles(folderName).ForEach(File.Delete)
```

edited Apr 21 '18 at 19:04

answered Jun 2 '10 at 22:43

zumalifeguard
6,565 5 23 37

3

```
private void ClearFolder(string FolderName)
{
    DirectoryInfo dir = new DirectoryInfo(FolderName);

    foreach (FileInfo fi in dir.GetFiles())
    {
        fi.IsReadOnly = false;
        fi.Delete();
    }

    foreach (DirectoryInfo di in dir.GetDirectories())
    {
        ClearFolder(di.FullName);
        di.Delete();
    }
}
```

edited Nov 28 '16 at 13:28

Mong Zhu
15.7k 5 25 50

answered Jul 27 '16 at 10:34

user6606429

1

It's not the best way to deal with the issue above. But it's an alternative one...

```
while (Directory.GetDirectories(dirpath).Length > 0)
 {
       //Delete all files in directory
       while (Directory.GetFiles(Directory.GetDirectories(dirpath)[0
       {
            File.Delete(Directory.GetFiles(dirpath)[0]);
       }
       Directory.Delete(Directory.GetDirectories(dirpath)[0]);
 }
```

edited Jul 26 '16 at 14:18

answered Jul 26 '16 at 14:12

dsmyrnaios

115 2 11

1

The following example shows how you can do that. It first creates some directories and a file and then removes them via `Directory.Delete(topPath, true);` :

```
    static void Main(string[] args)
    {
        string topPath = @"C:\NewDirectory";
        string subPath = @"C:\NewDirectory\NewSubDirectory";
```

```
        try
        {
            Directory.CreateDirectory(subPath);

            using (StreamWriter writer = File.CreateText(subPath + @
            {
                writer.WriteLine("content added");
            }

            Directory.Delete(topPath, true);

            bool directoryExists = Directory.Exists(topPath);

            Console.WriteLine("top-level directory exists: " + direc
        }
        catch (Exception e)
        {
            Console.WriteLine("The process failed: {0}", e.Message);
        }
    }
```

It is taken from https://msdn.microsoft.com/en-us/library/fxeahc5f(v=vs.110).aspx.

edited Jul 25 '16 at 15:01

Matt
15.9k 7 71 114

answered Jul 25 '16 at 12:02

Salma Tofaily
11 1

31

The simplest way:

```
Directory.Delete(path,true);
Directory.CreateDirectory(path);
```

Be aware that this may wipe out some permissions on the folder.

edited Jul 24 '16 at 21:58

Brian Webster
20.7k 40 130 208

answered Aug 16 '15 at 8:43

Igor Mukhachev
434 4 4

8 be aware that this will remove any special permissions the path had – Matthew Lock Oct 27 '15 at 0:42

4 You need to add timeout between those two actions. try to run this code and you will get Exception: while (true) { Directory.Delete(@"C:\Myfolder", true); Directory.CreateDirectory(@"C:\Myfolder"); } – RcMan Mar 22 '17 at 13:46

66

The following code will clear the folder recursively:

```
private void clearFolder(string FolderName)
{
    DirectoryInfo dir = new DirectoryInfo(FolderName);

    foreach(FileInfo fi in dir.GetFiles())
    {
        fi.Delete();
    }

    foreach (DirectoryInfo di in dir.GetDirectories())
    {
        clearFolder(di.FullName);
        di.Delete();
    }
}
```

edited Jul 18 '16 at 11:03

Tshilidzi Mudau

2,891 23 34

answered May 4 '10 at 15:36

hiteshbiblog

661 5 2

Worked for me, while *Directory.Delete(path,true);* throwed complaining that the folder was not empy – Jack Griffin Dec 18 '16 at 9:27

Elegant use of recursion. Well done! – Tsar Bomba Jan 9 '17 at 19:29

0

To delete the folder, this is code using Text box and a button `using System.IO;` :

```
private void Deletebt_Click(object sender, EventArgs e)
{
    System.IO.DirectoryInfo myDirInfo = new DirectoryInfo(@"" + dele

    foreach (FileInfo file in myDirInfo.GetFiles())
    {
       file.Delete();
    }
    foreach (DirectoryInfo dir in myDirInfo.GetDirectories())
    {
       dir.Delete(true);
    }
}
```

edited Jul 18 '16 at 9:46

Tshilidzi Mudau

2,891 23 34

answered Jul 2 '15 at 9:32

Abdelrhman Khalil

1 1

0

```
foreach (string file in System.IO.Directory.GetFiles(path))
{
   System.IO.File.Delete(file);
}

foreach (string subDirectory in System.IO.Directory.GetDirectories(
{
    System.IO.Directory.Delete(subDirectory,true);
}
```

edited Jul 18 '16 at 9:17

Tshilidzi Mudau

2,891 23 34

answered Oct 26 '12 at 5:55

Manish Y

11 1

0

Call from main

```
static void Main(string[] args)
{
   string Filepathe =<Your path>
   DeleteDirectory(System.IO.Directory.GetParent(Filepathe).FullName
}
```

Add this method

```
public static void DeleteDirectory(string path)
{
    if (Directory.Exists(path))
    {
        //Delete all files from the Directory
```

```
        foreach (string file in Directory.GetFiles(path))
        {
            File.Delete(file);
        }
        //Delete all child Directories
        foreach (string directory in Directory.GetDirectories(path))
        {
             DeleteDirectory(directory);
        }
        //Delete a Directory
        Directory.Delete(path);
    }
}
```

edited Jul 18 '16 at 8:42

Tshilidzi Mudau
2,891 23 34

answered Apr 5 '16 at 14:15

sansalk
2,961 23 24

37

```
new System.IO.DirectoryInfo(@"C:\Temp").Delete(true);

//Or

System.IO.Directory.Delete(@"C:\Temp", true);
```

edited Jun 24 '16 at 6:40

answered Jul 18 '12 at 11:55

Thulasiram
7,062 6 36 40

1 The second option, Directory.Delete(String, Boolean) worked for me. – Stephen MacDougall Mar 4 '13 at 15:16

13 This deletes the root directory, where the OP specifically asked that it be retained. – Marc L. May 30 '14 at 17:30

2 `Delete` will throw if the directory doesn't exist, so it would be safer to do a `Directory.Exists` check first. – James Apr 24 '15 at 15:26

1 @James `Directory.Exists` is not enough; after the check, another thread may have renamed or removed the directory. It is safer to `try-catch` . – andre_ss6 Jan 19 '16 at 3:36

2 @Marv Careful with simply adding a `Directory.Create` because the recursive `Directory.Delete` is unfortunately not guaranteed to be synchronous.. – Andrew Hanlon Dec 10 '16 at 19:55

12

```
System.IO.Directory.Delete(installPath, true);
System.IO.Directory.CreateDirectory(installPath);
```

answered Dec 23 '15 at 8:53

MacGyver
11k 34 129 218

1 short, and to the point – Drew Dec 23 '15 at 8:57

3 same as above: be aware that this will remove any special permissions the path had. – hB0 May 31 '16 at 12:32

4

Using just static methods with File and Directory instead of FileInfo and DirectoryInfo will perform faster. (see accepted answer at What is the difference between File and FileInfo in C#?). Answer shown as utility method.

```
public static void Empty(string directory)
{
    foreach(string fileToDelete in System.IO.Directory.GetFiles(dire
    {
        System.IO.File.Delete(fileToDelete);
    }
    foreach(string subDirectoryToDeleteToDelete in
System.IO.Directory.GetDirectories(directory))
    {
        System.IO.Directory.Delete(subDirectoryToDeleteToDelete, tru
    }
}
```

edited May 23 '17 at 11:55

Community ♦

1 1

answered Dec 7 '15 at 19:22

Kriil

263 2 12

0

```
using System.IO;

string[] filePaths = Directory.GetFiles(@"c:\MyDir\");

foreach (string filePath in filePaths)

File.Delete(filePath);
```

edited Nov 30 '15 at 17:58

Julian E.

3,927 5 24 44

answered Nov 30 '15 at 17:11

SynsMasTer

11 4

2

This version does not use recursive calls, and solves the readonly problem.

```
public static void EmptyDirectory(string directory)
{
    // First delete all the files, making sure they are not readonly
    var stackA = new Stack<DirectoryInfo>();
    stackA.Push(new DirectoryInfo(directory));

    var stackB = new Stack<DirectoryInfo>();
    while (stackA.Any())
    {
        var dir = stackA.Pop();
        foreach (var file in dir.GetFiles())
        {
            file.IsReadOnly = false;
            file.Delete();
        }
        foreach (var subDir in dir.GetDirectories())
        {
            stackA.Push(subDir);
            stackB.Push(subDir);
        }
    }

    // Then delete the sub directories depth first
    while (stackB.Any())
    {
        stackB.Pop().Delete();
    }
}
```

answered Oct 23 '15 at 8:26

Jeppe Andreasen

371 3 3

0

this will show how we delete the folder and check for it we use Text box

```
using System.IO;
namespace delete_the_folder
{
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void Deletebt_Click(object sender, EventArgs e)
    {
        //the  first you should write the folder place
        if (Pathfolder.Text=="")
        {
            MessageBox.Show("ples write the path of the folder");
            Pathfolder.Select();
            //return;
        }

        FileAttributes attr = File.GetAttributes(@Pathfolder.Text);

        if (attr.HasFlag(FileAttributes.Directory))
            MessageBox.Show("Its a directory");
        else
            MessageBox.Show("Its a file");

        string path = Pathfolder.Text;
        FileInfo myfileinf = new FileInfo(path);
        myfileinf.Delete();

    }


}

}
```

edited Jun 30 '15 at 8:44

Robert

4,075 12 53 108

answered Jun 30 '15 at 8:38

Abdelrhman Khalil
1 1

1

use DirectoryInfo's GetDirectories method.

```
foreach (DirectoryInfo subDir in new DirectoryInfo(targetDir).GetDir
                    subDir.Delete(true);
```

answered May 27 '15 at 18:57

Mr_Hmp
1,614 1 24 39

6

The following code will clean the directory, but leave the root directory there (recursive).

```
Action<string> DelPath = null;
DelPath = p =>
{
    Directory.EnumerateFiles(p).ToList().ForEach(File.Delete);
    Directory.EnumerateDirectories(p).ToList().ForEach(DelPath);
    Directory.EnumerateDirectories(p).ToList().ForEach(Directory.Del
};
DelPath(path);
```

edited Mar 23 '15 at 17:02

Peter Mortensen
13.9k 19 87 114

answered Jul 30 '14 at 11:49

hofi
101 1 3

3

In Windows 7, if you have just created it manually with Windows Explorer, the directory structure is similar to this one:

```
C:
  \AAA
    \BBB
      \CCC
        \DDD
```

And running the code suggested in the original question to clean the directory C:\AAA, the line `di.Delete(true)` always fails with IOException "The directory is not empty" when trying to delete BBB. It is probably because of some kind of delays/caching in Windows Explorer.

The following code works reliably for me:

```
static void Main(string[] args)
{
    DirectoryInfo di = new DirectoryInfo(@"c:\aaa");
    CleanDirectory(di);
}

private static void CleanDirectory(DirectoryInfo di)
{
    if (di == null)
        return;

    foreach (FileSystemInfo fsEntry in di.GetFileSystemInfos())
    {
        CleanDirectory(fsEntry as DirectoryInfo);
        fsEntry.Delete();
    }
    WaitForDirectoryToBecomeEmpty(di);
}
```

```
private static void WaitForDirectoryToBecomeEmpty(DirectoryInfo di)
{
    for (int i = 0; i < 5; i++)
    {
        if (di.GetFileSystemInfos().Length == 0)
            return;
        Console.WriteLine(di.FullName + i);
        Thread.Sleep(50 * i);
    }
}
```

edited Mar 23 '15 at 17:01

Peter Mortensen
13.9k 19 87 114

answered Sep 22 '12 at 13:45

farfareast
1,589 1 17 19

What's is about stackoverflow.com/questions/12415105/… "When you call Directory.Delete and a file is open in such way, Directory.Delete succeeds in deleting all files but when Directory.Delete calls RemoveDirectory a "directory is not empty" exception is thrown because there is a file marked for deletion but not actually deleted." – Kiquenet Jul 19 '13 at 7:55

@Kiquenet: Looks like we found an issue in Windows. Windows could have consulted the list of files marked for deletion and if all files in the directory are marked for deletion, do not say that directory is not empty. Anyway my WaitForDirectoryToBecomeEmpty() is a workaround. – farfareast Jul 26 '13 at 16:56

37

We can also show love for LINQ:

```
using System.IO;
using System.Linq;
…
```

```
var directory = Directory.GetParent(TestContext.TestDir);

directory.EnumerateFiles()
    .ToList().ForEach(f => f.Delete());

directory.EnumerateDirectories()
    .ToList().ForEach(d => d.Delete(true));
```

Note that my solution here is not performant, because I am using `Get*().ToList().ForEach(...)` which generates the same `IEnumerable` twice. I use an extension method to avoid this issue:

```
using System.IO;
using System.Linq;
…
var directory = Directory.GetParent(TestContext.TestDir);

directory.EnumerateFiles()
    .ForEachInEnumerable(f => f.Delete());

directory.EnumerateDirectories()
    .ForEachInEnumerable(d => d.Delete(true));
```

This is the extension method:

```
/// <summary>
/// Extensions for <see cref="System.Collections.Generic.IEnumerable
/// </summary>
public static class IEnumerableOfTExtensions
{
    /// <summary>
    /// Performs the <see cref="System.Action"/>
    /// on each item in the enumerable object.
    /// </summary>
    /// <typeparam name="TEnumerable">The type of the enumerable.</ty
    /// <param name="enumerable">The enumerable.</param>
    /// <param name="action">The action.</param>
    /// <remarks>
    /// “I am philosophically opposed to providing such a method, for
    /// …The first reason is that doing so violates the functional pr
principles
    /// that all the other sequence operators are based upon. Clearly
of a call
```

```
    /// to this method is to cause side effects.”
    /// –Eric Lippert, “foreach” vs “ForEach”
[http://blogs.msdn.com/b/ericlippert/archive/2009/05/18/foreach-vs-f
    /// </remarks>
    public static void ForEachInEnumerable<TEnumerable>(this IEnumer
enumerable, Action<TEnumerable> action)
    {
        foreach (var item in enumerable)
        {
            action(item);
        }
    }
}
```

edited Mar 23 '15 at 16:59

Peter Mortensen
13.9k 19 87 114

answered Aug 11 '10 at 20:00

rasx
3,442 1 35 55

1 And if you're trying to delete subdirectories as well, `foreach (var dir in info.GetDirectories("*", SearchOption.AllDirectories).OrderByDescending(dir => dir.FullName.Length)) dir.Delete();` might be of use. – Warty Jan 1 '14 at 1:31

1 If you like performance, consider using `directory.EnumerateFiles()` and `directory.EnumerateDirectories()` instead of the `directory.Get*()` methods. – Tinister Apr 15 '14 at 17:22

1 Funny, my own `IEnumerable<T>.ForEach()` extension has a summary XML comment, "Violation! Violation! Unclean!". – Marc L. Jun 2 '14 at 19:58

Hey whats the 2nd reason? The 3rd? Etc.? – flaZer Mar 31 '17 at 21:17

lol @RASX - he's talking to you: "If you don’t agree with these philosophical objections and find practical value in this pattern, by all means, go ahead and write this trivial one-liner yourself." – flaZer Mar 31 '17 at 21:24

0

```
using System;
using System.IO;
namespace DeleteFoldersAndFilesInDirectory
{
     class Program
     {
          public static void DeleteAll(string path)
          {
               string[] directories = Directory.GetDirectories(path)
               string[] files = Directory.GetFiles(path);
               foreach (string x in directories)
                    Directory.Delete(x, true);
               foreach (string x in files)
                    File.Delete(x);
          }
          static void Main()
          {
               Console.WriteLine("Enter The Directory:");
               string directory = Console.ReadLine();
               Console.WriteLine("Deleting all files and directories
               DeleteAll(directory);
               Console.WriteLine("Deleted");
          }
     }
}
```

answered May 30 '14 at 17:08

Diaa Eddin

89 6

-2

```
private void ClearDirectory(string path)
{
    if (Directory.Exists(path))//if folder exists
    {
        Directory.Delete(path, true);//recursive delete (all subdirs
    }
```

```
    Directory.CreateDirectory(path);//creates empty directory
}
```

answered Feb 11 '14 at 8:15

dadziu
9 1

2 See below..."deleting and recreating" is not the same as keeping, all ACL customizations will be lost. – Marc L. May 30 '14 at 17:28

I've tried something very similar to this since I didn't care about and ACL customizations and ran into issues with the folder not being created after `Directory.CreateDirectory` – JG in SD Feb 16 '17 at 20:55

-3

The only thing you should do is to set `optional recursive parameter` to `True` .

```
Directory.Delete("C:\MyDummyDirectory", True)
```

Thanks to .NET. :)

answered Dec 10 '13 at 16:09

LysanderM
39 1 3

3 This also deletes the directory itself. – rajat Dec 20 '13 at 7:22

163

Yes, that's the correct way to do it. If you're looking to give yourself a "Clean" (or, as I'd prefer to call it, "Empty" function), you can create an extension method.

```
public static void Empty(this System.IO.DirectoryInfo directory)
{
    foreach(System.IO.FileInfo file in directory.GetFiles()) file.De
    foreach(System.IO.DirectoryInfo subDirectory in directory.GetDir
subDirectory.Delete(true);
}
```

This will then allow you to do something like..

```
System.IO.DirectoryInfo directory = new System.IO.DirectoryInfo(@"C:

directory.Empty();
```

edited Jul 22 '13 at 11:44

answered Aug 17 '09 at 15:58

Adam Robinson
150k 27 256 322

4 The last line should be subDirectory.Delete(true) instead of directory.Delete(true). I just cut-and-pasted the code and it deleted the main directory itself. Thanks for the code it's great! – Aximili Jun 9 '10 at 5:18

24 note that `Empty` exists in C# already, for `string`. If I saw something else named `Empty` I would be surprised if it modified the object (or filesystem) instead of giving me a `bool` that says if it is empty or not. Because of that, I would go with the name `Clean`. – Default May 24 '12 at 7:14

5 @Default: I don't think the fact that one type has a property already should have any bearing on whether another (completely unrelated) type should have it; and the convention for properties and functions that indicate state for words that can also be verbs is to prefix them with `Is` (i.e. `IsEmpty` rather than `Empty`). – Adam Robinson May 24 '12 at 14:09

3 @AdamRobinson Just wanted to make note of it. To *me*, what Microsoft has in their code do have some bearing. But it's for everyone to interpret

:) – Default May 24 '12 at 14:20

3 @simonhaines: The point of the question was to *empty* the directory (i.e. delete everything *inside of it*), not to delete the directory itself. – Adam Robinson Sep 26 '13 at 3:58

-4

```
IO.Directory.Delete(HttpContext.Current.Server.MapPath(path), True)
```

You don't need more than that

edited Jul 16 '13 at 20:03

Anri

5,127 3 28 56

answered Jul 16 '13 at 19:45

Mustafa Odeh

1 1

2 Wrong... this will also delete the root directory. – L-Four Aug 24 '13 at 12:03

0

```
DirectoryInfo Folder = new DirectoryInfo(Server.MapPath(path));
if (Folder .Exists)
{
    foreach (FileInfo fl in Folder .GetFiles())
    {
        fl.Delete();
    }

    Folder .Delete();
}
```

edited Apr 11 '13 at 7:32

Entreco
9,967 6 62 80

answered Apr 11 '13 at 7:13

Ashok Luhach
1 1

Could you be more specific and explain how and why this should work? – Deep Frozen Apr 11 '13 at 7:32

3 Answers with only code are not suitable. You should explain how and why it should work/solve the problem. – rdurand Apr 11 '13 at 7:37

7

Every method that I tried, they have failed at some point with System.IO errors. The following method works for sure, even if the folder is empty or not, read-only or not, etc.

```
ProcessStartInfo Info = new ProcessStartInfo();
Info.Arguments = "/C rd /s /q \"C:\\MyFolder"";
Info.WindowStyle = ProcessWindowStyle.Hidden;
Info.CreateNoWindow = true;
Info.FileName = "cmd.exe";
Process.Start(Info);
```

answered Oct 11 '12 at 10:06

Alexandru Dicu
637 8 22

1 I always prefer rd /s /q + mkdir when it comes to emptying directories. – Hadi Di Wao May 21 '14 at 16:23

7 This is not cross-platform solution. Unix-like systems clearly don't have cmd.exe, they don't even run .exe files. C# is not Windows only, there's also Mono, which is cross-platform. – Sarge Borsch May 2 '15 at 8:15

@SargeBorsch there was no cross platform requirement in the question and being C# it's most likely the solution will be used for Windows. It seems to be the only answer which doesn't use .NET functions, so it's pretty valuable as an alternative. – Alex Pandrea Apr 3 at 10:27

2

```
string directoryPath = "C:\Temp";
Directory.GetFiles(directoryPath).ToList().ForEach(File.Delete);
Directory.GetDirectories(directoryPath).ToList().ForEach(Directory.D
```

answered Jun 14 '12 at 17:41

AVH

1,135 9 7

An exception of type 'System.IO.IOException' occurred in mscorlib.dll but was not handled in user code Additional information: The directory is not empty. – kipusoep Jul 22 '16 at 7:28

protected by Community ♦ Mar 4 '17 at 0:24

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?