



MICROSOFT ARCHITECT

By Joydip Kanjilal, Columnist, InfoWorld | MAR 28, 2017 10:05 AM PDT

How to work with FileSystemWatcher in C#

The FileSystemWatcher class can be used to monitor changes to file system and trigger events when such changes occur



Insight Cloud + Data Center Transformation Read the report	Forrester report on pay-as-you-go on-premises A new approach lets you better control costs and reduce risk. Click to watch video	Converged and Hyper-converged Infrastructure Driving value for modern organizations. View infographic now	CI v. HCI: How they differ Learn more about the architecture, features, and benefits of each model.
---	--	---	--

```
1 reference
private static void MonitorDirectory(string path)
{
    FileSystemWatcher fileSystemWatcher = new FileSystemWatcher();
    fileSystemWatcher.Path = path;
    fileSystemWatcher.Created += FileSystemWatcher_Created;
    fileSystemWatcher.Renamed += FileSystemWatcher_Renamed;
    fileSystemWatcher.Deleted += FileSystemWatcher_Deleted;
    fileSystemWatcher.EnableRaisingEvents = true; // Starts monitoring
}

1 reference
private static void FileSystemWatcher_Created(object sender, FileSystemEventArgs e)
1 reference
private static void FileSystemWatcher_Renamed(object sender, FileSystemEventArgs e)
1 reference
private static void FileSystemWatcher_Deleted(object sender, FileSystemEventArgs e)
```



The FileSystemWatcher class in the System.IO namespace can be used to monitor changes to the file system. It watches a file or a directory in your system for changes and triggers events when changes occur.

In order for the FileSystemWatcher to work, you should specify a directory that needs to be monitored. The FileSystemWatcher raises the following events when changes occur to a directory that it is monitoring.

- Changed: This event is triggered when a file or a directory in the path being monitored is changed
- Created: This event is triggered when a file or a directory in the path being monitored is created
- Deleted: This event is triggered when a file or a directory in the path being monitored is deleted
- Error: This event is triggered there is an error due to changes made in the path being monitored
- Renamed: This event is triggered when a file or a directory in the path being monitored is renamed

Creating a simple file system watcher in C#

Let's create a new console application project in Visual Studio to demonstrate how a typical file system watcher works. Note that a better way to use the FileSystemWatcher class would be by using a Windows Service. You can build a Windows Service that uses the FileSystemWatcher class and sends out notifications as and when changes occur to the path being watched.

[Also on InfoWorld: [How to build gRPC applications in ASP.NET Core](#)]

Anyway, let's now get into a bit of code now. In the Main method of the




Anyway, let's now get into a bit of code now. In the main method of the Program.cs file, write the following code.


```
static void Main(string[] args)


{

    string path = @"D:\IDG";
```

RECOMMENDED WHITEPAPERS

 **Cloud Analytics: Accelerate BI and Maximize ROI with Smart OLAP™**

 **Accelerate Your BI on Trillions of Rows | Cloud | Big Data**

 **Do More with Less: Use APIs to Drive Usage of Existing Resources**

```
MonitorDirectory(path);
```



SponsoredPost Sponsored by SAP Digital Interconnect
What I Learned From Working as an Intrapreneur
I discovered that adopting an intrapreneurial mindset helped me address customer needs, grow the business, motivate my team, and above all, keep my sanity.

```
Console.ReadKey();
```

```
}
```

The following code snippet shows how the MonitorDirectory method would look like. This method would be used to monitor a particular directory and raise events whenever a change occurs. The directory path is passed as an argument to the method.

```
private static void MonitorDirectory(string path)
```



SponsoredPost Sponsored by SAP
Communications Platform as-a-Service Finds its Inflection Point
CPaaS solutions can eliminate connectivity and device challenges, reduce app integration costs, and ease IT support responsibilities.

```
{

    FileSystemWatcher fileSystemWatcher = new
FileSystemWatcher();

    fileSystemWatcher.Path = path;

    fileSystemWatcher.Created +=
FileSystemWatcher_Created;

    fileSystemWatcher.Renamed +=
FileSystemWatcher_Renamed;

    fileSystemWatcher.Deleted +=
FileSystemWatcher_Deleted;

    fileSystemWatcher.EnableRaisingEvents = true;

}
```

Note how the events are declared and that the EnableRaisingEvents

NOTE how the events are declared and that the `EnableRaisingEvents` property of the file system watcher object is set to true to enable raising events when a change on the path being monitored occurs. In essence, this starts the actual monitoring -- you are informing `FileSystemWatcher` to start monitoring the path and raise appropriate events henceforth.

For each of the events that you have declared, you should have the respective event handler that gets executed when the event is triggered. Here's the source code of the event handlers that would be triggered as and when a change to the directory being monitored occurs.

```
private static void FileSystemWatcher_Created(object sender,
FileSystemEventArgs e)
{
    Console.WriteLine("File created: {0}", e.Name);
}

private static void FileSystemWatcher_Renamed(object
sender, FileSystemEventArgs e)
{
    Console.WriteLine("File renamed: {0}", e.Name);
}

private static void FileSystemWatcher_Deleted(object
sender, FileSystemEventArgs e)
{
    Console.WriteLine("File deleted: {0}", e.Name);
}
```

Here's the complete source code for your reference.

```
using System;

using System.IO;

namespace IDGFileSystemWatcher
{
    class Program
    {
        static void Main(string[] args)
        {
            string path = @"D:\IDG";

            MonitorDirectory(path);

            Console.ReadKey();
        }

        private static void MonitorDirectory(string path)
        {
            FileSystemWatcher fileSystemWatcher = new
            FileSystemWatcher();

            fileSystemWatcher.Path = path;

            fileSystemWatcher.Created +=
            FileSystemWatcher_Created;
```

```

FileSystemWatcher_Created,

        fileSystemWatcher.Renamed +=
FileSystemWatcher_Renamed;

        fileSystemWatcher.Deleted +=
FileSystemWatcher_Deleted;

        fileSystemWatcher.EnableRaisingEvents = true;
    }

    private static void FileSystemWatcher_Created(object
sender, FileSystemEventArgs e)
    {
        Console.WriteLine("File created: {0}", e.Name);
    }

    private static void FileSystemWatcher_Renamed(object
sender, FileSystemEventArgs e)
    {
        Console.WriteLine("File renamed: {0}", e.Name);
    }

    private static void FileSystemWatcher_Deleted(object
sender, FileSystemEventArgs e)
    {
        Console.WriteLine("File deleted: {0}", e.Name);
    }
}
}

```

Assuming that the directory named IDG is available on the D:\> drive of your system, run the console application and then create a new file in the IDG directory. You would observe that the name of the newly created file is displayed in the console window. This is because as soon as a new file is created in the directory being monitored (D:\IDG in our example), the FileSystemWatcher_Created event is triggered.

Related: [C#](#) [Software Development](#)

Joydip Kanjilal is a Microsoft MVP in ASP.Net, as well as a speaker and author of several books and articles. He has more than 20 years of experience in IT including more than 16 years in Microsoft .Net and related technologies.

Follow [!\[\]\(e78f798d4ea5c530c9db49e7d26e6b95_img.jpg\)](#) [!\[\]\(034433b90593e82e5460e34e3ed48e9b_img.jpg\)](#) [!\[\]\(5f24500834b50a8307ffe63e419281a9_img.jpg\)](#) [!\[\]\(8502790a2fc8970abb55aa7f7a7a6bae_img.jpg\)](#) [!\[\]\(7f7375e819602f97f5594a28879b3e09_img.jpg\)](#)

Copyright © 2017 IDG Communications, Inc.

- Stay up to date with InfoWorld's newsletters for software developers, analysts, database programmers, and data scientists.
- Get expert insights from our member-only Insider articles.

SPONSORED LINKS

Join the IDG TECH(talk) Community, an exclusive online network where IT experts find resources to enhance their knowledge and career.




Digital Transformation wasn't supposed to happen this way. You need visibility to gain control. Take control with NETSCOUT .

dtSearch® instantly searches terabytes of files, emails, databases, web data. See site for hundreds of reviews; enterprise & developer evaluations

Software defines your networks. NETSCOUT defines your visibility. See it all.

A network flooded with new connections can't afford an unwelcome one. Find the DDoS Threat with NETSCOUT.

InfoWorld
FROM IDG

FOLLOW US   

[ABOUT US](#) [CONTACT](#) [PRIVACY POLICY](#) [COOKIE POLICY](#) [MEMBER PREFERENCES](#) [ADVERTISING](#) [IDG CAREERS](#) [AD CHOICES](#) [E-COMMERCE LINKS](#)
[CALIFORNIA: DO NOT SELL MY PERSONAL INFO](#)

 Copyright © 2020 IDG Communications, Inc.

Explore the IDG Network ▼