

Forum:[Hack Forums](#)

Thread: [FileSystemWatcher disposing.](#)

severos - September 11, 2015, 4:31 am

I'm writing a small class that will keep an eye on certain folders for changes, and backup the ones that changes periodically, to monitor the folder for changes I used FileSystemWatcher with some events handlers to add changed files to a list.

Code `FileSystemWatcher watcher = new FileSystemWatcher(source, "");`
`watcher.IncludeSubdirectories = true;`
`watcher.Disposed += DisposedWatcher;` Now I want to delete those files from the list when the watcher object is disposed, so I used a Disposed handler
Code `private void DisposedWatcher(object source, EventArgs e)` source and e has those methods: ToString/Equals/GetType/GetHashCode.

My question is, how would I know what was the path I'm watching?? and what exactly is the object source at first place? I see it in all events but I don't know what it is.

Thanks in advance.

Odus - September 11, 2015, 4:40 am

Declare your FSW at class level. Path property should be what you're looking for.

severos - September 11, 2015, 5:23 am

(Sep 11, 2015 04:40 AM)Odus Wrote: [\[CITE\]](#) Declare your FSW at class level. Path property should be what you're looking for.

Thanks for the tip, but not exactly what I was looking for.?
I already have the FSW objects in a list and calling the Dispose method on them.
Looks like all I had to do is a simple casting for the object source to turn it back into a FSW object and I can get the Path property out of it and compare it with paths of files in list.

Nikey646 - September 11, 2015, 4:03 pm

(Sep 11, 2015 05:23 AM)severos Wrote: [\[CITE\]](#) Thanks for the tip, but not exactly what I was looking for.?

I already have the FSW objects in a list and calling the Dispose method on them.
Looks like all I had to do is a simple casting for the object source to turn it back into a FSW object and I can get the Path property out of it and compare it with paths of files in list. Your "Source" object is typically called "sender", and usually contains the class that created the event.

For example, create a new void like this:

Code `private void EventExample(object sender, EventArgs e)`
`{ }` and if you were to call it in the "public Form1" void (Constructor) like this:
Code `EventExample(this, EventArgs.Empty);` Then it would provide you with an Empty EventArgs, and the sender would be the instance of Form1 that made the request. From which you could cast it back to Form1, ensure it's what you are expecting, and move along with what you wanted to do.

The biggest problem you are going to encounter when you attempt to get the FileSystemWatcher from the Disposed event, is that it will likely be disposed already....As such, all data retaining to the FSW will be gone. If you really must have something done when the FSW is being disposed, the only way i can think of doing it would be to extend the class, override the .Dispose method, and then call base.Dispose(); at the end of your method, allowing the FSW to complete it's disposal.

