.NET Framework 類別庫

FileSystemWatcher 類別

當目錄或目錄內的檔案變更時，接聽 (Listen) 檔案系統變更告知並引發事件。

命名空間： System.IO
組件： System (在 system.dll 中)

語法

Visual Basic (宣告)

```
Public Class FileSystemWatcher
    Inherits Component
    Implements ISupportInitialize
```

Visual Basic (使用方式)

```
Dim instance As FileSystemWatcher
```

C#

```
public class FileSystemWatcher : Component, ISupportInitialize
```

C++

```
public ref class FileSystemWatcher : public Component, ISupportInitialize
```

J#

```
public class FileSystemWatcher extends Component implements ISupportInitialize
```

JScript

```
public class FileSystemWatcher extends Component implements ISupportInitialize
```

備註

使用 **FileSystemWatcher** 監看指定目錄內的變更。您可以監看指定目錄內檔案和子目錄的變更。您可以建立元件，在本機電腦 (Local Machine)、網路磁碟機或遠端電腦上監看檔案。

若要監看所有具有副檔名之檔案中的變更，請將 Filter [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.filter(VS.80).aspx] 屬性設爲空字串 ("")，或使用萬用字元 ("*.*")。若要監看指定的檔案，請設定 **Filter** 屬性爲檔案名稱。例如，要監看 MyDoc.txt 檔案中的變更，請設定 **Filter** 屬性爲 "MyDoc.txt"。您也可以監看某個類型檔案中的變更。例如，要監看文字檔中的變更，請設定 **Filter** 屬性爲 "*.txt"。

在目錄或檔案中，您可以監看許多類型的變更。例如，您可以監看檔案或目錄的 **Attributes**、**LastWrite** 日期和時間或 **Size** 的變更。這可以設定 NotifyFilter [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.notifyfilter(VS.80).aspx] 屬性爲其中一個 NotifyFilters [http://msdn.microsoft.com/zh-tw/library/system.io.notifyfilters(VS.80).aspx] 值來完成。如需您所能監看的變更類型之詳細資訊，請參閱 **NotifyFilters**。

您可以監看檔案或目錄的重新命名、刪除或建立。例如，若要監看文字檔的重新命名，可將 **Filter** 屬性設定爲 "*.txt"，並呼叫指定了 Renamed [http://msdn.microsoft.com/zh-tw/library/system.io.watcherchangetypes.renamed(VS.80).aspx] 做爲其參數的 WaitForChanged [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.waitforchanged(VS.80).aspx] 方法。

Windows 作業系統會在 **FileSystemWatcher** 所建立的緩衝區中，告知您的元件有檔案變更。如果在短時間內有很多的變更，緩衝區會溢位。這樣會造成元件失去追蹤目錄中變更的線索，而它將只會提供概括性的告知。使用 InternalBufferSize [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.internalbuffersize(VS.80).aspx] 屬性增加緩衝區大小會高度耗費資源，因爲它來自不能交換到磁碟的未分頁記憶體，所以盡可能讓緩衝區愈小愈好，但是不至於小到會遺漏任何檔案變更事件。若要避免緩衝區溢位，使用 **NotifyFilter** 和 IncludeSubdirectories

[http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.includesubdirectories(VS.80).aspx] 屬性，您就可以篩選掉不想要的變更告知。

如需 **FileSystemWatcher** 執行個體的初始屬性值清單，請參閱 FileSystemWatcher [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.filesystemwatcher(VS.80).aspx] 建構函式。

使用 **FileSystemWatcher** 類別 (Class) 時，請注意下列項目。

- 隱藏檔案不會被略過。
- 在一些系統中，**FileSystemWatcher** 會使用簡短 8.3 檔案名稱格式報告檔案的變更。例如，變更爲 "LongFileName.LongExtension" 可能會報告爲 "LongFi~.Lon"。
- 此類別包含了適用於所有成員之類別層級的連結要求和繼承 (Inheritance) 要求。當立即呼叫端或衍生類別 (Derived Class) 不具備完全信任的使用權限時，就會擲回 SecurityException [http://msdn.microsoft.com/zh-tw/library/system.security.securityexception(VS.80).aspx] 。如需安全性需求的詳細資訊，請參閱連結要求 [http://msdn.microsoft.com/zh-tw/library/hzsc022c(VS.80).aspx] 。

複製和移動資料夾

作業系統和 **FileSystemWatcher** 物件會將剪貼動作或移動動作解譯爲資料夾和其內容的重新命名動作。如果您將具有檔案的資料夾剪貼至所監看的資料夾，**FileSystemWatcher** 物件僅會將該資料夾而不是將其內容做爲新項目報告，因爲僅是將它們重新命名。

若要在資料夾的內容移動或複製至監看的資料夾中時進行告知，請提供 OnChanged [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.onchanged(VS.80).aspx] 和 OnRenamed [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.onrenamed(VS.80).aspx] 事件處理常式方法，如下表中所示。

事件處理常式	處理的事件	執行
OnChanged	Changed [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.changed(VS.80).aspx] , Created [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.created(VS.80).aspx] , Deleted [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.deleted(VS.80).aspx]	報告檔案屬性中的變更，建立的檔案和刪除的檔案。
OnRenamed	Renamed [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.renamed(VS.80).aspx]	列出重新命名之檔案和資料夾的新舊路徑，可按需要遞迴地展開。

事件和緩衝區大小

請注意，數個因素會影響有哪些檔案系統變更事件會被引發，如以下所述：

- 通用檔案系統作業可能會引發一個以上的事件。例如，當將檔案從一個目錄移至另一個目錄時，可能會引發幾個 **OnChanged** 以及一些 OnCreated [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.oncreated(VS.80).aspx] 和 OnDeleted [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.ondeleted(VS.80).aspx] 事件。移動檔案是一個包含多個簡單作業的複雜作業，因此會引發多個事件。相同的，某些應用程式 (例如防毒軟體) 可能會造成 **FileSystemWatcher** 所偵測到之其他的檔案系統事件。
- **FileSystemWatcher** 可以監看磁碟，只要它們沒有被切換或移除。**FileSystemWatcher** 沒有引發 CD 和 DVD 的事件，因爲時間戳記和屬性無法變更。遠端電腦必須安裝其中一個必要的平台，以便元件能夠正常運作。然而，您不能從 Windows NT 4.0 電腦監看遠端的 Windows NT 4.0 電腦。
- 如果有多個 **FileSystemWatcher** 物件在 Windows XP Service Pack 1 之前版本或 Windows 2000 SP2 (含) 以前版本中監看相同 UNC 路徑，則只有其中一個物件會引發事件。在執行 Windows XP SP1 及較新版本、Windows 2000 SP3 及較新版本，或 Windows Server 2003 的電腦上，所有 **FileSystemWatcher** 物件都會引發適當的事件。

- 設定 **Filter** 並不會減少進入緩衝區的資料數量。

請注意，當遺漏事件或超過緩衝區大小時，由於與 Windows 作業系統的相依性，**FileSystemWatcher** 並不會引發 Error [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.error(VS.80).aspx] 事件。爲了避免遺漏事件，請遵循下列方針：

- 使用 **InternalBufferSize** 屬性增加緩衝區大小，可以防止遺漏檔案系統變更事件。
- 避免監看具有長檔名的檔案，考慮使用較短名稱重新命名檔案。
- 盡可能讓事件處理程式碼越短越好。

範例

下列範例會建立 **FileSystemWatcher**，以監看在執行階段所指定的目錄。元件設定成監看目錄中文字檔的 **LastWrite** 和 **LastAccess** 時間、建立、刪除或重新命名的變更。如果變更、建立或刪除檔案，檔案路徑會列印到主控台。將檔案重新命名時，新舊路徑會列印到主控台。

在這個範例中使用 System.Diagnostics [http://msdn.microsoft.com/zh-tw/library/system.diagnostics(VS.80).aspx] 和 System.IO [http://msdn.microsoft.com/zh-tw/library/system.io(VS.80).aspx] 命名空間。

Visual Basic 複製程式碼

```
Public Class Watcher

    Public Shared Sub Main()

         Run()

    End Sub

    <PermissionSet(SecurityAction.Demand, Name:="FullTrust")> _
    Private Shared Sub Run

      Dim args() As String = System.Environment.GetCommandLineArgs()
        ' If a directory is not specified, exit the program.
        If args.Length <> 2 Then
            ' Display the proper way to call the program.
            Console.WriteLine("Usage: Watcher.exe (directory)")
            Return
        End If

        ' Create a new FileSystemWatcher and set its properties.
        Dim watcher As New FileSystemWatcher()
        watcher.Path = args(1)
        ' Watch for changes in LastAccess and LastWrite times, and
        ' the renaming of files or directories.
        watcher.NotifyFilter = (NotifyFilters.LastAccess Or NotifyFilters.LastWrite Or
NotifyFilters.FileName Or NotifyFilters.DirectoryName)
        ' Only watch text files.
        watcher.Filter = "*.txt"

        ' Add event handlers.
        AddHandler watcher.Changed, AddressOf OnChanged
        AddHandler watcher.Created, AddressOf OnChanged
        AddHandler watcher.Deleted, AddressOf OnChanged
        AddHandler watcher.Renamed, AddressOf OnRenamed

        ' Begin watching.
        watcher.EnableRaisingEvents = True

        ' Wait for the user to quit the program.
        Console.WriteLine("Press 'q' to quit the sample.")
        While Chr(Console.Read()) <> "q"c
        End While
    End Sub

    ' Define the event handlers.
    Private Shared Sub OnChanged(source As Object, e As FileSystemEventArgs)
        ' Specify what is done when a file is changed, created, or deleted.
        Console.WriteLine("File: " & e.FullPath & " " & e.ChangeType)
    End Sub

    Private Shared Sub OnRenamed(source As Object, e As RenamedEventArgs)
```

```
        ' Specify what is done when a file is renamed.
        Console.WriteLine("File: {0} renamed to {1}", e.OldFullPath, e.FullPath)
    End Sub

End Class
```

C# 複製程式碼

```
public class Watcher
{

    public static void Main()
    {
    Run();

    }

    [PermissionSet(SecurityAction.Demand, Name="FullTrust")]
    public static void Run()
    {
        string[] args = System.Environment.GetCommandLineArgs();

        // If a directory is not specified, exit program.
        if(args.Length != 2)
        {
            // Display the proper way to call the program.
            Console.WriteLine("Usage: Watcher.exe (directory)");
            return;
        }

        // Create a new FileSystemWatcher and set its properties.
        FileSystemWatcher watcher = new FileSystemWatcher();
        watcher.Path = args[1];
        /* Watch for changes in LastAccess and LastWrite times, and
           the renaming of files or directories. */
        watcher.NotifyFilter = NotifyFilters.LastAccess | NotifyFilters.LastWrite
           | NotifyFilters.FileName | NotifyFilters.DirectoryName;
        // Only watch text files.
        watcher.Filter = "*.txt";

        // Add event handlers.
        watcher.Changed += new FileSystemEventHandler(OnChanged);
        watcher.Created += new FileSystemEventHandler(OnChanged);
        watcher.Deleted += new FileSystemEventHandler(OnChanged);
        watcher.Renamed += new RenamedEventHandler(OnRenamed);

        // Begin watching.
        watcher.EnableRaisingEvents = true;

        // Wait for the user to quit the program.
        Console.WriteLine("Press \'q\' to quit the sample.");
        while(Console.Read()!='q');
    }

    // Define the event handlers.
    private static void OnChanged(object source, FileSystemEventArgs e)
    {
        // Specify what is done when a file is changed, created, or deleted.
       Console.WriteLine("File: " +  e.FullPath + " " + e.ChangeType);
    }

    private static void OnRenamed(object source, RenamedEventArgs e)
    {
        // Specify what is done when a file is renamed.
        Console.WriteLine("File: {0} renamed to {1}", e.OldFullPath, e.FullPath);
    }
}
```

C++ 複製程式碼

```
public ref class Watcher
{
private:
   // Define the event handlers.
   static void OnChanged( Object^ /*source*/, FileSystemEventArgs^ e )
   {
      // Specify what is done when a file is changed, created, or deleted.
      Console::WriteLine( "File: {0} {1}", e->FullPath, e->ChangeType );
```

```
    }

    static void OnRenamed( Object^ /*source*/, RenamedEventArgs^ e )
    {
       // Specify what is done when a file is renamed.
       Console::WriteLine( "File: {0} renamed to {1}", e->OldFullPath, e->FullPath );
    }

public:
    [PermissionSet(SecurityAction::Demand, Name="FullTrust")]
    int static run()
    {
       array<String^>^args = System::Environment::GetCommandLineArgs();

       // If a directory is not specified, exit program.
       if ( args->Length != 2 )
       {
          // Display the proper way to call the program.
          Console::WriteLine( "Usage: Watcher.exe (directory)" );
          return 0;
       }

       // Create a new FileSystemWatcher and set its properties.
       FileSystemWatcher^ watcher = gcnew FileSystemWatcher;
       watcher->Path = args[ 1 ];

       /* Watch for changes in LastAccess and LastWrite times, and
           the renaming of files or directories. */
       watcher->NotifyFilter = static_cast<NotifyFilters>(NotifyFilters::LastAccess |
             NotifyFilters::LastWrite | NotifyFilters::FileName | NotifyFilters::DirectoryName);

       // Only watch text files.
       watcher->Filter = "*.txt";

       // Add event handlers.
       watcher->Changed += gcnew FileSystemEventHandler( Watcher::OnChanged );
       watcher->Created += gcnew FileSystemEventHandler( Watcher::OnChanged );
       watcher->Deleted += gcnew FileSystemEventHandler( Watcher::OnChanged );
       watcher->Renamed += gcnew RenamedEventHandler( Watcher::OnRenamed );

       // Begin watching.
       watcher->EnableRaisingEvents = true;

       // Wait for the user to quit the program.
       Console::WriteLine( "Press \'q\' to quit the sample." );
       while ( Console::Read() != 'q' )
          ;
    }
};

int main() {
   Watcher::run();
}
```

J# 複製程式碼

```
public class Watcher
{
    public static void main(String[] args1)
    {
    Run();
    }

    /** @attribute PermissionSet(SecurityAction.Demand, Name="FullTrust")
     */
    public static void Run()
    {
        String args[] = System.Environment.GetCommandLineArgs();

        // If a directory is not specified, exit program.
        if (args.length != 2) {

            // Display the proper way to call the program.
            Console.WriteLine("Usage: Watcher.exe (directory)");
            return;
        }

        // Create a new FileSystemWatcher and set its properties.
        FileSystemWatcher watcher = new FileSystemWatcher();
```

```
        watcher.set_Path(args[1]);

        /* Watch for changes in LastAccess and LastWrite times, and
           the renaming of files or directories.
         */
        watcher.set_NotifyFilter
            (NotifyFilters.LastAccess |NotifyFilters.LastWrite |
            NotifyFilters.FileName | NotifyFilters.DirectoryName);

        // Only watch text files.
        watcher.set_Filter("*.txt");

        // Add event handlers.
        watcher.add_Changed(new FileSystemEventHandler(OnChanged));
        watcher.add_Created(new FileSystemEventHandler(OnChanged));
        watcher.add_Deleted(new FileSystemEventHandler(OnChanged));
        watcher.add_Renamed(new RenamedEventHandler(OnRenamed));

        // Begin watching.
        watcher.set_EnableRaisingEvents(true);

        // Wait for the user to quit the program.
        Console.WriteLine("Press \'q\' to quit the sample.");
        while ((Console.Read() != 'q')) {

        }
    }

    // Define the event handlers.
    private static void OnChanged(Object source,FileSystemEventArgs e)
    {
        // Specify what is done when a file is changed, created, or deleted.
        Console.WriteLine(("File: " + e.get_FullPath() + " "
            + e.get_ChangeType()));
    } //OnChanged

    private static void OnRenamed(Object source, RenamedEventArgs e)
    {
        // Specify what is done when a file is renamed.
        Console.WriteLine("File: {0} renamed to {1}",
            e.get_OldFullPath(),e.get_FullPath());
    } //OnRenamed
} //Watcher
```

.NET Framework 安全性

- SecurityPermission [http://msdn.microsoft.com/zh-tw/library/system.security.permissions.securitypermission(VS.80).aspx] 用來呼叫 ProcessStartInfo [http://msdn.microsoft.com/zh-tw/library/system.diagnostics.processstartinfo(VS.80).aspx] 的成員。要求值：LinkDemand [http://msdn.microsoft.com/zh-tw/library/system.security.permissions.securityaction.linkdemand(VS.80).aspx] ，具名使用權限集合：**FullTrust**。
- **SecurityPermission** 用於衍生自 **ProcessStartInfo** 類別。要求值：InheritanceDemand [http://msdn.microsoft.com/zh-tw/library/system.security.permissions.securityaction.inheritancedemand(VS.80).aspx] ，具名使用權限集合：**FullTrust**。

繼承階層架構

System.Object [http://msdn.microsoft.com/zh-tw/library/system.object(VS.80).aspx]
  System.MarshalByRefObject [http://msdn.microsoft.com/zh-tw/library/system.marshalbyrefobject(VS.80).aspx]
    System.ComponentModel.Component [http://msdn.microsoft.com/zh-tw/library/system.componentmodel.component(VS.80).aspx]
      System.IO.FileSystemWatcher

執行緒安全

這個型別的所有公用靜態成員 (即 Visual Basic 中的 **Shared** 成員) 都是安全執行緒。並非所有的執行個體成員均爲安全執行緒。

平台

Windows 98、 Windows 2000 SP4、 Windows Server 2003、 Windows XP Media Center Edition、 Windows XP Professional x64 Edition、 Windows XP SP2、 Windows XP Starter Edition

.NET Framework 並不支援各種平台的所有版本。如需支援平台版本的相關資訊，請參閱系統需求 [http://msdn.microsoft.com/zh-tw/library/8z6watww(VS.80).aspx] 一節的內容。

版本資訊

.NET Framework

支援版本：2.0、1.1、1.0

請參閱

參考

FileSystemWatcher 成員 [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher_members(VS.80).aspx]
System.IO 命名空間 [http://msdn.microsoft.com/zh-tw/library/system.io(VS.80).aspx]
FileSystemWatcher.NotifyFilter [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.notifyfilter(VS.80).aspx]
NotifyFilters [http://msdn.microsoft.com/zh-tw/library/system.io.notifyfilters(VS.80).aspx]
FileSystemEventArgs 類別 [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemeventargs(VS.80).aspx]
FileSystemEventHandler 委派 [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemeventhandler(VS.80).aspx]
Filter [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.filter(VS.80).aspx]
IncludeSubdirectories [http://msdn.microsoft.com/zh-tw/library/system.io.filesystemwatcher.includesubdirectories(VS.80).aspx]
InternalBufferOverflowException [http://msdn.microsoft.com/zh-tw/library/system.io.internalbufferoverflowexception(VS.80).aspx]
RenamedEventArgs [http://msdn.microsoft.com/zh-tw/library/system.io.renamedeventargs(VS.80).aspx]
RenamedEventHandler [http://msdn.microsoft.com/zh-tw/library/system.io.renamedeventhandler(VS.80).aspx]
WaitForChangedResult [http://msdn.microsoft.com/zh-tw/library/system.io.waitforchangedresult(VS.80).aspx]
WatcherChangeTypes [http://msdn.microsoft.com/zh-tw/library/t6xf43e0(VS.80).aspx]

標記:

社群內容