DirectoryInfo.GetFiles Method

Reference

Definition

Namespace: System.IO

Assembly: System.Runtime.dll

Returns a file list from the current directory.

Overloads

Expand table

GetFiles(String, EnumerationOptions)	Returns a file list from the current directory matching the specified search pattern and enumeration options.
GetFiles(String, SearchOption)	Returns a file list from the current directory matching the given search pattern and using a value to determine whether to search subdirectories.
GetFiles()	Returns a file list from the current directory.
GetFiles(String)	Returns a file list from the current directory matching the given search pattern.

GetFiles(String, EnumerationOptions)

Source: DirectoryInfo.cs

Returns a file list from the current directory matching the specified search pattern and enumeration options.

C#

```
public System.IO.FileInfo[] GetFiles (string searchPattern,
System.IO.EnumerationOptions enumerationOptions);
```

Parameters

searchPattern String

The search string to match against the names of files. This parameter can contain a combination of valid literal path and wildcard (* and ?) characters, but it doesn't support regular expressions.

`enumerationOptions` EnumerationOptions

An object that describes the search and enumeration configuration to use.

Returns

FileInfo[]

An array of strongly typed FileInfo objects that match `searchPattern` and `enumerationOptions`.

Exceptions

ArgumentException

.NET Framework and .NET Core versions older than 2.1: `searchPattern` contains one or more invalid characters defined by the GetInvalidPathChars() method.

ArgumentNullException

`searchPattern` is `null`.

DirectoryNotFoundException

The path is invalid (for example, it is on an unmapped drive).

SecurityException

The caller does not have the required permission.

Remarks

`searchPattern` can be a combination of literal and wildcard characters, but it doesn't support regular expressions. The following wildcard specifiers are permitted in `searchPattern`.

Expand table

Wildcard specifier	Matches
* (asterisk)	Zero or more characters in that position.
? (question mark)	Zero or one character in that position.

Characters other than the wildcard are literal characters. For example, the string "*t" searches for all names in ending with the letter "t". The `searchPattern` string "s*" searches for all names in `path` beginning with the letter "s".

The EnumerateFiles and GetFiles methods differ as follows:

- When you use EnumerateFiles, you can start enumerating the collection of FileInfo objects before the whole collection is returned.
- When you use GetFiles, you must wait for the whole array of FileInfo objects to be returned before you can access the array.

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

If there are no files in the DirectoryInfo, this method returns an empty array.

The following wildcard specifiers are permitted in the `searchPattern` parameter.

Expand table

Wildcard character	Description
*	Zero or more characters.
?	Exactly zero or one character.

The order of the returned file names is not guaranteed; use the Sort method if a specific sort order is required.

Wildcards are permitted. For example, the `searchPattern` string "*.txt" searches for all file names having an extension of "txt". The `searchPattern` string "s*" searches for all file names beginning with the letter "s". If there are no files, or no files that match the `searchPattern` string in the DirectoryInfo, this method returns an empty array.

> **Note**
>
> When using the asterisk wildcard character in a `searchPattern` (for example, "*.txt"), the matching behavior varies depending on the length of the specified file extension. A `searchPattern` with a file extension of exactly three characters returns files with an extension of three or more characters, where the first three characters match the file extension specified in the `searchPattern`. A `searchPattern` with a file extension of one, two, or more than three characters

returns only files with extensions of exactly that length that match the file extension specified in the `searchPattern`. When using the question mark wildcard character, this method returns only files that match the specified file extension. For example, given two files in a directory, "file1.txt" and "file1.txtother", a search pattern of "file?.txt" returns only the first file, while a search pattern of "file*.txt" returns both files.

> **Note**
>
> Because this method checks against file names with both the 8.3 file name format and the long file name format, a search pattern similar to "*1*.txt" may return unexpected file names. For example, using a search pattern of "*1*.txt" will return "longfilename.txt" because the equivalent 8.3 file name format would be "longf~1.txt".

This method pre-populates the values of the following FileInfo properties:

- Attributes
- CreationTime
- CreationTimeUtc
- LastAccessTime
- LastAccessTimeUtc
- LastWriteTime
- LastWriteTimeUtc
- Length

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Standard	2.1

GetFiles(String, SearchOption)

Source: DirectoryInfo.cs

Returns a file list from the current directory matching the given search pattern and using a value to determine whether to search subdirectories.

```
C#

public System.IO.FileInfo[] GetFiles (string searchPattern,
System.IO.SearchOption searchOption);
```

Parameters

`searchPattern` String

The search string to match against the names of files. This parameter can contain a combination of valid literal path and wildcard (* and ?) characters, but it doesn't support regular expressions.

`searchOption` SearchOption

One of the enumeration values that specifies whether the search operation should include only the current directory or all subdirectories.

Returns

FileInfo[]

An array of type FileInfo.

Exceptions

ArgumentException

.NET Framework and .NET Core versions older than 2.1: `searchPattern` contains one or more invalid characters defined by the GetInvalidPathChars() method.

ArgumentNullException

`searchPattern` is `null`.

ArgumentOutOfRangeException

`searchOption` is not a valid SearchOption value.

DirectoryNotFoundException

The path is invalid (for example, it is on an unmapped drive).

SecurityException

The caller does not have the required permission.

Examples

The following example shows how to get a list of files from a directory by using different search options. The example assumes a directory that has files named log1.txt, log2.txt, test1.txt, test2.txt, test3.txt, and a subdirectory that has a file named SubFile.txt.

C#

```
using System;
using System.IO;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            DirectoryInfo di = new DirectoryInfo(@"C:\Users\tom-
fitz\Documents\ExampleDir");
            Console.WriteLine("No search pattern returns:");
            foreach (var fi in di.GetFiles())
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();

            Console.WriteLine("Search pattern *2* returns:");
            foreach (var fi in di.GetFiles("*2*"))
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();

            Console.WriteLine("Search pattern test?.txt
returns:");
            foreach (var fi in di.GetFiles("test?.txt"))
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();
```

```
            Console.WriteLine("Search pattern AllDirectories re-
turns:");
            foreach (var fi in di.GetFiles("*",
SearchOption.AllDirectories))
            {
                Console.WriteLine(fi.Name);
            }
        }
    }
}
/*
This code produces output similar to the following:

No search pattern returns:
log1.txt
log2.txt
test1.txt
test2.txt
test3.txt

Search pattern *2* returns:
log2.txt
test2.txt

Search pattern test?.txt returns:
test1.txt
test2.txt
test3.txt

Search pattern AllDirectories returns:
log1.txt
log2.txt
test1.txt
test2.txt
test3.txt
SubFile.txt
Press any key to continue . . .

*/
```

Remarks

The EnumerateFiles and GetFiles methods differ as follows:

- When you use EnumerateFiles, you can start enumerating the collection of FileInfo objects before the whole collection is returned.
- When you use GetFiles, you must wait for the whole array of FileInfo objects to be returned before you can access the array.

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

If there are no files in the DirectoryInfo, this method returns an empty array.

The following wildcard specifiers are permitted in `searchPattern`.

Expand table

Wildcard character	Description
* (asterisk)	Zero or more characters.
? (question mark)	Exactly zero or one character.

The order of the returned file names is not guaranteed; use the Sort method if a specific sort order is required.

Wildcards are permitted. For example, the `searchPattern` string "*.txt" searches for all file names having an extension of "txt". The `searchPattern` string "s*" searches for all file names beginning with the letter "s". If there are no files, or no files that match the `searchPattern` string in the DirectoryInfo, this method returns an empty array.

> **Note**
>
> When using the asterisk wildcard character in a `searchPattern` (for example, "*.txt"), the matching behavior varies depending on the length of the specified file extension. A `searchPattern` with a file extension of exactly three characters returns files with an extension of three or more characters, where the first three characters match the file extension specified in the `searchPattern`. A `searchPattern` with a file extension of one, two, or more than three characters returns only files with extensions of exactly that length that match the file extension specified in the `searchPattern`. When using the question mark wildcard character, this method returns only files that match the specified file extension. For example, given two files in a directory, "file1.txt" and "file1.txtother", a search pattern of "file?.txt" returns only the first file, while a search pattern of "file*.txt" returns both files.

The following list shows the behavior of different lengths for the `searchPattern` parameter:

- "*.abc" returns files having an extension of .abc, .abcd, .abcde, .abcdef, and so on.
- "*.abcd" returns only files having an extension of .abcd.
- "*.abcde" returns only files having an extension of .abcde.
- "*.abcdef" returns only files having an extension of .abcdef.

> **Note**
>
> Because this method checks against file names with both the 8.3 file name format and the long file name format, a search pattern similar to "*1*.txt" may return unexpected file names. For example, using a search pattern of "*1*.txt" will return "longfilename.txt" because the equivalent 8.3 file name format would be "longf~1.txt".

This method pre-populates the values of the following FileInfo properties:

1. Attributes
2. CreationTime
3. CreationTimeUtc
4. LastAccessTime
5. LastAccessTimeUtc
6. LastWriteTime
7. LastWriteTimeUtc
8. Length

See also

- File and Stream I/O
- How to: Read Text from a File
- How to: Write Text to a File

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.3, 1.4, 1.6, 2.0, 2.1
UWP	10.0

GetFiles()

Source: DirectoryInfo.cs

Returns a file list from the current directory.

C#

```
public System.IO.FileInfo[] GetFiles ();
```

Returns

FileInfo[]

An array of type FileInfo.

Exceptions

DirectoryNotFoundException
The path is invalid, such as being on an unmapped drive.

Examples

The following example shows how to get a list of files from a directory by using different search options. The example assumes a directory that has files named log1.txt, log2.txt, test1.txt, test2.txt, test3.txt, and a subdirectory that has a file named SubFile.txt.

C#

```
using System;
using System.IO;
```

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            DirectoryInfo di = new DirectoryInfo(@"C:\Users\tom-
fitz\Documents\ExampleDir");
            Console.WriteLine("No search pattern returns:");
            foreach (var fi in di.GetFiles())
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();

            Console.WriteLine("Search pattern *2* returns:");
            foreach (var fi in di.GetFiles("*2*"))
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();

            Console.WriteLine("Search pattern test?.txt
returns:");
            foreach (var fi in di.GetFiles("test?.txt"))
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();

            Console.WriteLine("Search pattern AllDirectories re-
turns:");
            foreach (var fi in di.GetFiles("*",
SearchOption.AllDirectories))
            {
                Console.WriteLine(fi.Name);
            }
        }
    }
}
/*
This code produces output similar to the following:

No search pattern returns:
log1.txt
log2.txt
test1.txt
test2.txt
test3.txt

Search pattern *2* returns:
```

```
log2.txt
test2.txt

Search pattern test?.txt returns:
test1.txt
test2.txt
test3.txt

Search pattern AllDirectories returns:
log1.txt
log2.txt
test1.txt
test2.txt
test3.txt
SubFile.txt
Press any key to continue . . .

*/
```

Remarks

The EnumerateFiles and GetFiles methods differ as follows:

- When you use EnumerateFiles, you can start enumerating the collection of FileInfo objects before the whole collection is returned.
- When you use GetFiles, you must wait for the whole array of FileInfo objects to be returned before you can access the array.

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

If there are no files in the DirectoryInfo, this method returns an empty array.

The order of the returned file names is not guaranteed; use the Sort method if a specific sort order is required.

This method pre-populates the values of the following FileInfo properties:

- Attributes
- CreationTime
- CreationTimeUtc
- LastAccessTime
- LastAccessTimeUtc

- LastWriteTime
- LastWriteTimeUtc
- Length

See also

- File and Stream I/O
- How to: Read Text from a File
- How to: Write Text to a File

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.3, 1.4, 1.6, 2.0, 2.1
UWP	10.0

GetFiles(String)

Source: DirectoryInfo.cs

Returns a file list from the current directory matching the given search pattern.

C#

```
public System.IO.FileInfo[] GetFiles (string searchPattern);
```

Parameters

`searchPattern` String

The search string to match against the names of files. This parameter can contain a combination of valid literal path and wildcard (* and ?) characters, but it doesn't

support regular expressions.

Returns

FileInfo[]

An array of type FileInfo.

Exceptions

ArgumentException

.NET Framework and .NET Core versions older than 2.1: `searchPattern` contains one or more invalid characters defined by the GetInvalidPathChars() method.

ArgumentNullException

`searchPattern` is `null`.

DirectoryNotFoundException

The path is invalid (for example, it is on an unmapped drive).

SecurityException

The caller does not have the required permission.

Examples

The following example shows how to get a list of files from a directory by using different search options. The example assumes a directory that has files named log1.txt, log2.txt, test1.txt, test2.txt, test3.txt, and a subdirectory that has a file named SubFile.txt.

C#

```
using System;
using System.IO;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            DirectoryInfo di = new DirectoryInfo(@"C:\Users\tom-
fitz\Documents\ExampleDir");
            Console.WriteLine("No search pattern returns:");
            foreach (var fi in di.GetFiles())
```

```
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();

            Console.WriteLine("Search pattern *2* returns:");
            foreach (var fi in di.GetFiles("*2*"))
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();

            Console.WriteLine("Search pattern test?.txt
returns:");
            foreach (var fi in di.GetFiles("test?.txt"))
            {
                Console.WriteLine(fi.Name);
            }

            Console.WriteLine();

            Console.WriteLine("Search pattern AllDirectories re-
turns:");
            foreach (var fi in di.GetFiles("*",
SearchOption.AllDirectories))
            {
                Console.WriteLine(fi.Name);
            }
        }
    }
}
/*
This code produces output similar to the following:

No search pattern returns:
log1.txt
log2.txt
test1.txt
test2.txt
test3.txt

Search pattern *2* returns:
log2.txt
test2.txt

Search pattern test?.txt returns:
test1.txt
test2.txt
test3.txt

Search pattern AllDirectories returns:
log1.txt
log2.txt
```

```
test1.txt
test2.txt
test3.txt
SubFile.txt
Press any key to continue . . .

*/
```

Remarks

`searchPattern` can be a combination of literal and wildcard characters, but it doesn't support regular expressions. The following wildcard specifiers are permitted in `searchPattern`.

Expand table

Wildcard specifier	Matches
* (asterisk)	Zero or more characters in that position.
? (question mark)	Zero or one character in that position.

Characters other than the wildcard are literal characters. For example, the string "*t" searches for all names in ending with the letter "t". The `searchPattern` string "s*" searches for all names in `path` beginning with the letter "s".

The EnumerateFiles and GetFiles methods differ as follows:

- When you use EnumerateFiles, you can start enumerating the collection of FileInfo objects before the whole collection is returned.
- When you use GetFiles, you must wait for the whole array of FileInfo objects to be returned before you can access the array.

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

If there are no files in the DirectoryInfo, this method returns an empty array.

The following wildcard specifiers are permitted in the `searchPattern` parameter.

Expand table

Wildcard character	Description
*	Zero or more characters.
?	Exactly zero or one character.

The order of the returned file names is not guaranteed; use the Sort method if a specific sort order is required.

Wildcards are permitted. For example, the `searchPattern` string "*.txt" searches for all file names having an extension of "txt". The `searchPattern` string "s*" searches for all file names beginning with the letter "s". If there are no files, or no files that match the `searchPattern` string in the DirectoryInfo, this method returns an empty array.

> **Note**
>
> When using the asterisk wildcard character in a `searchPattern` (for example, "*.txt"), the matching behavior varies depending on the length of the specified file extension. A `searchPattern` with a file extension of exactly three characters returns files with an extension of three or more characters, where the first three characters match the file extension specified in the `searchPattern`. A `searchPattern` with a file extension of one, two, or more than three characters returns only files with extensions of exactly that length that match the file extension specified in the `searchPattern`. When using the question mark wildcard character, this method returns only files that match the specified file extension. For example, given two files in a directory, "file1.txt" and "file1.txtother", a search pattern of "file?.txt" returns only the first file, while a search pattern of "file*.txt" returns both files.

> **Note**
>
> Because this method checks against file names with both the 8.3 file name format and the long file name format, a search pattern similar to "*1*.txt" may return unexpected file names. For example, using a search pattern of "*1*.txt" will return "longfilename.txt" because the equivalent 8.3 file name format would be "longf~1.txt".

This method pre-populates the values of the following FileInfo properties:

- Attributes

- CreationTime
- CreationTimeUtc
- LastAccessTime
- LastAccessTimeUtc
- LastWriteTime
- LastWriteTimeUtc
- Length

See also

- File and Stream I/O
- How to: Read Text from a File
- How to: Write Text to a File

Applies to

▼ .NET 9 and other versions

Product	Versions
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7, 8, 9
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
.NET Standard	1.3, 1.4, 1.6, 2.0, 2.1
UWP	10.0

more information, see our contributor guide.