Work with file paths in .NET

5 minutes

100 XP

.NET has a built-in mechanism for working with file-system paths.

In the previous exercise, we didn't have many folders to parse. If you have a file system with many files and folders, manually building paths can be tedious. Thankfully, .NET provides some built-in constants and utility functions to make it easier to handle file paths.

Here, you learn about some of the constants and utility functions in the `System.IO` namespace and the `System.Environment` type so that you can make your program smarter and more resilient.

Determine the current directory

Sometimes, you don't know which directory or path your program should run in. Let's say you want your program to use the current directory, but you don't know the path.

.NET exposes the full path to the current directory via the `Directory.GetCurrentDirectory` method.

C#

Copy

```
Console.WriteLine(Directory.GetCurrentDirectory());
```

If you run the above code from the *201* folder in the following structure, `Directory.GetCurrentDirectory()` returns `stores\201`:

Copy

```
stores
  201
```

Work with special directories

.NET runs everywhere: on Windows, macOS, Linux, and even on mobile operating systems like iOS and Android. Each operating system may or may not have the concept of special system folders. Such as, a home directory, which is dedicated for user-specific files, or a desktop directory, or a directory for storing temporary files.

Those types of special directories differ for each operating system. It would be cumbersome to try to remember each operating system's directory structure and perform switches based on the current OS.

The `System.Environment.SpecialFolder` enumeration specifies constants to retrieve paths to special system folders.

The following code returns the path to the equivalent of the Windows *My Documents* folder, or the user's *HOME* directory for any operating system, even if the code is running on Linux:

C#

Copy

```
string docPath = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
```

Work with paths

Paths are a subject that comes up so frequently that .NET includes a class called `Path` specifically for working with them.

The `Path` class is located in the `System.IO` namespace of .NET, and doesn't need to be installed.

Special path characters

Different operating systems use different characters to separate directory levels.

For example, Windows uses the backslash (`stores\201`) and macOS uses the forward slash (`stores/201`).

To help you use the correct character, the `Path` class contains the `DirectorySeparatorChar` field.

.NET automatically interprets that field into the separator character that's applicable to the operating system when you need to build a path manually.

C#

Copy

```
Console.WriteLine($"stores{Path.DirectorySeparatorChar}201");

// returns:
// stores\201 on Windows
//
// stores/201 on macOS
```

Join paths

The `Path` class works with the concept of file and folder paths, which are just strings. You can use the `Path` class to automatically build correct paths for specific operating systems.

For instance, if you want to get the path to the *stores/201* folder, you can use the `Path.Combine` function to do that.

C# Copy

```
Console.WriteLine(Path.Combine("stores","201")); // outputs: stores/201
```

Remember, you should use the `Path.Combine` or `Path.DirectorySeparatorChar` class instead of hard-coding strings, because your program might be running on many different operating systems. The `Path` class always formats the paths correctly for the operating system it's running on.

> **Tip**
>
> The `Path` class doesn't care whether things actually exist. Paths are conceptual, not physical, and the class is building and parsing strings for you.

Determine filename extensions

The `Path` class can also tell you the extension of a filename. If you have a file and you want to know if it's a JSON file, you can use the `Path.GetExtension` function.

C# Copy

```
Console.WriteLine(Path.GetExtension("sales.json")); // outputs: .json
```

Get everything you need to know about a file or path

The `Path` class contains many different methods that do various things. You can get the most information about a directory or a file by using the `DirectoryInfo` or `FileInfo` classes, respectively.

C# Copy

```
string fileName = $"stores{Path.DirectorySeparatorChar}201{Path.DirectorySeparatorChar}sales{Path.DirectorySeparatorChar}sales.json";

FileInfo info = new FileInfo(fileName);

Console.WriteLine($"Full Name: {info.FullName}{Environment.NewLine}Directory: {info.Directory}{Environment.NewLine}Extension: {info.Extension}
```

There are many more useful properties and utility methods on the `Path`, `DirectoryInfo`, and `FileInfo` classes, but these core concepts are the ones that you're likely to use most often. In the next exercise, you'll compose paths and identify .json files.