Work with the file system

3 minutes

200 XP

.NET contains built-in types for working with the file system that you can find in the `System.IO` namespace.

Here, you learn about the types available in `System.IO` by using C# to read a file system to discover files and directories.

The scenario

Large retailers often write data to files so it can be processed later in batches.

Tailwind Traders has each of its stores write its sales total to a file and send that file to a central location. To use those files, the company needs to create a batch process that can work with the file system.

Include the System.IO namespace

The System.IO namespace contains built-in types that allow you to interact with files and directories. For example, you can retrieve collections of files and directories based on search criteria and `get` and `set` properties for files and directories. You can also use `System.IO` namespace types to synchronously and asynchronously read and write data streams and files.

For now, we're going to focus on what you need to know to work with directories by using the `Directory` class contained in the `System.IO` namespace. The Directory class exposes static methods for creating, moving, and enumerating through directories and subdirectories.

List all directories

The `Directory` class is often used to list out (or *enumerate*) directories. For instance, the Tailwind Traders file structure has a root folder called *stores*. In that folder are subfolders organized by store number, and inside those folders are the sales-total and inventory files. The structure looks like this example:

Copy

```
📁 stores
    📄 sales.json
    📄 totals.txt
    📁 201
       📄 sales.json
       📄 salestotals.json
       📄 inventory.txt
    📁 202
```

To read through and list the names of the top-level directories, use the `Directory.EnumerateDirectories` function.

C#

Copy

```csharp
IEnumerable<string> listOfDirectories = Directory.EnumerateDirectories("stores");

foreach (var dir in listOfDirectories) {
    Console.WriteLine(dir);
}

// Outputs:
// stores/201
// stores/202
```

List files in a specific directory

To list the names of all of the files in a directory, you can use the `Directory.EnumerateFiles` function.

C#

Copy

```csharp
IEnumerable<string> files = Directory.EnumerateFiles("stores");

foreach (var file in files)
{
    Console.WriteLine(file);
}
```

```
// Outputs:
// stores/totals.txt
// stores/sales.json
```

List all content in a directory and all subdirectories

Both the `Directory.EnumerateDirectories` and `Directory.EnumerateFiles` functions have an overload that accepts a parameter to specify that search pattern files and directories must match.

They also have another overload that accepts a parameter to indicate whether to recursively traverse a specified folder and all of its subfolders.

C# Copy

```csharp
// Find all *.txt files in the stores folder and its subfolders
IEnumerable<string> allFilesInAllFolders = Directory.EnumerateFiles("stores", "*.txt", SearchOption.AllDirectories);

foreach (var file in allFilesInAllFolders)
{
    Console.WriteLine(file);
}

// Outputs:
// stores/totals.txt
// stores/201/inventory.txt
```

In the next exercise, you use the `Directory` class to dynamically read through Tailwind Traders' main *stores* directory to find all of the sales.json files.

Check your knowledge

1. Which of the following tasks can you do with Directory.EnumerateFiles?

List the files of a directory and subdirectories.

You can use Directory.EnumerateFiles to recursively enumerate all files in a directory and subdirectories that match a certain pattern.

List the subdirectories of a directory.

Write files to a directory.

Next unit: Exercise - Work with the file system