Exercise - Work with the file system

5 minutes

100 XP

You can use .NET to find and return information about files and folders.

Tailwind Traders has many physical stores all over the world. Every night, each store creates a file named *sales.json* that contains the total of all sales for that day. These files are organized in folders named with the store ID.

Note

This module uses the .NET CLI (Command Line Interface) and Visual Studio Code for local development. After completing this module, you can apply the concepts you've learned by using a development environment like Visual Studio (Windows) or Visual Studio for Mac (macOS), or continue development in Visual Studio Code (Windows, Linux, & macOS).

This module uses the .NET 6.0 SDK. Ensure that you have .NET 6.0 installed by running the following command in your preferred terminal:

.NET CLI

Copy

```
dotnet --list-sdks
```

Output similar to the following appears:

Console

Copy

```
3.1.100 [C:\program files\dotnet\sdk]
5.0.100 [C:\program files\dotnet\sdk]
6.0.100 [C:\program files\dotnet\sdk]
```

Ensure that a version that starts with `6` is listed. If none is listed or the command isn't found, install the most recent .NET 6.0 SDK.

Clone the project

In this exercise, you write a .NET program that searches a directory and its subdirectories for files named *sales.json*.

A starter project has already been created for you. You clone it using the integrated terminal in Visual Studio Code.

1. Open Visual Studio Code.
2. In the main menu, select **View** > **Terminal** to open a TERMINAL window.
3. (Optional) In the TERMINAL window, change to a directory you want to copy the files to, such as `c:\MyProjects`.
4. In the TERMINAL window, run the following command to clone the starter project and go to the cloned project:

   Bash

   Copy

   ```
   git clone https://github.com/MicrosoftDocs/mslearn-dotnet-files && cd mslearn-dotnet-files
   ```

5. Run the following command to create a new .NET Console project:

   Bash

   Copy

   ```
   dotnet new console -f net6.0 -n mslearn-dotnet-files -o .
   ```

6. Run the following command to open the new .NET project in the same instance of Visual Studio Code:

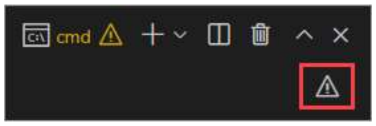
   Bash

   Copy

   ```
   code -a .
   ```

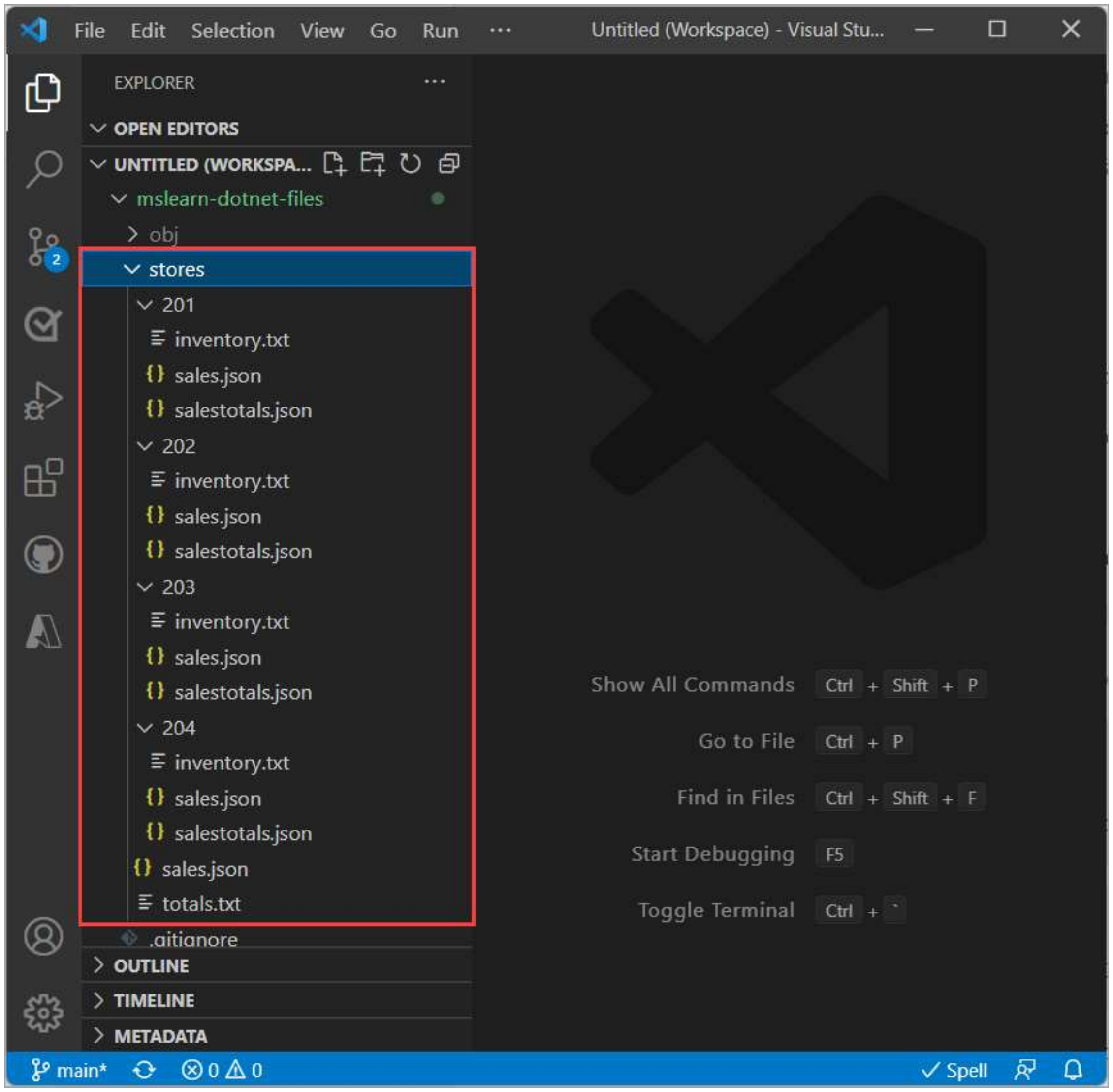
   Tip

   At this point, Visual Studio Code may prompt you that required assets to build and run the project are missing.



Select the triangle with the exclamation point and then select **Relaunch terminal** to add the files that allow Visual Studio Code to run and debug the project.

7. In the EXPLORER window, under **mslearn-dotnet-files**, expand the **stores** folder and each of the numbered folders inside.

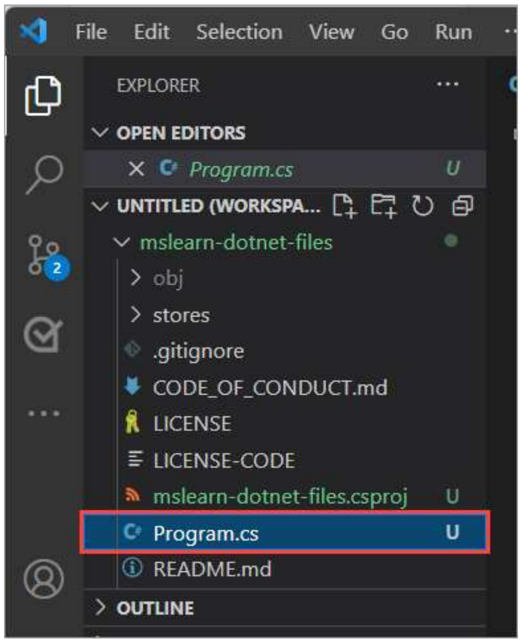


Find the sales.json files

The following tasks create a program to find all the sales.json files in all folders of the `mslearn-dotnet-files` project.

Include the System.IO namespace

1. In the EXPLORER window, select the `Program.cs` file to open it in the editor.



2. Paste the following code into the first line of the `Program.cs` file to import the `System.IO` and `System.Collections.Generic` namespaces:

C# Copy

```
using System.IO;
using System.Collections.Generic;
```

Note

Starting with .NET 6, the two statements in the above code are automatically included in a new project by way of the `ImplcitUsings` property group. Because we specified the `-f net6.0` flag when we created a new console project, they are implicitly added. However, if you're working with an older project, they need to be included in the `Program.cs` file, and it doesn't affect this project if you leave them in the file.

Write a function to find the sales.json files

Create a new function called `FindFiles` that takes a `folderName` parameter.

1. Replace the **Console.WriteLine("Hello, World!");** line with the following code:

C# Copy

```
IEnumerable<string> FindFiles(string folderName)
{
    List<string> salesFiles = new List<string>();

    var foundFiles = Directory.EnumerateFiles(folderName, "*", SearchOption.AllDirectories);

    foreach (var file in foundFiles)
    {
        // The file name will contain the full path, so only check the end of it
        if (file.EndsWith("sales.json"))
        {
            salesFiles.Add(file);
        }
    }

    return salesFiles;
}
```

2. Insert the following code below the `using` statements to call the `FindFiles` function. This code passes in the *stores* folder name as the location to search for files.

C# Copy

```
var salesFiles = FindFiles("stores");
```

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

3. Press `Ctrl+S` (or `Cmd+S` macOS) to save the `Program.cs` file.

Run the program

1. Enter the following command in the terminal window to run the program:

   Bash

   Copy

   ```bash
   dotnet run
   ```

2. The program should show the following output:

   Output

   Copy

   ```
   stores/sales.json
   stores/201/sales.json
   stores/202/sales.json
   stores/203/sales.json
   stores/204/sales.json
   ```

Excellent! You've successfully written a command-line program that traverses all folders in the `stores` directory and lists all the *sales.json* files that were found.

In this example, the path to the *stores* directory was rather simple, and in the working directory of the program. In the next unit, you'll learn how to construct complex structures that work across operating systems by using the `Path` class.

Got stuck?

If you had any problems running the program, here's the completed code for the `Program.cs` file. Replace the contents of your `Program.cs` file with this code:

C#

Copy

```csharp
var salesFiles = FindFiles("stores");

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

IEnumerable<string> FindFiles(string folderName)
{
    List<string> salesFiles = new List<string>();

    var foundFiles = Directory.EnumerateFiles(folderName, "*", SearchOption.AllDirectories);

    foreach (var file in foundFiles)
    {
        // The file name will contain the full path, so only check the end of it
        if (file.EndsWith("sales.json"))
        {
            salesFiles.Add(file);
        }
    }

    return salesFiles;
}
```