補充教材

檔案與串流 (FILE I/O)

HORAZON

C#程式設計

本章目標

1. 了解 System.IO 命名空間
2. 學會 **讀取 (Read)** 與 **寫入 (Write)** 文字檔 (.txt)
3. 認識 File, Directory, Path 類別
4. 實作：極簡日記系統

為什麼需要檔案存取？

變數存在 **記憶體 (RAM)** 中，程式關掉就不見了。
為了讓資料永久保存，我們需要將資料寫入 **硬碟 (Disk)**。

常見應用

- 遊戲存檔
- 錯誤日誌 (Error Logs)
- 設定檔 (Config)
- 匯出報表

SYSTEM.IO

C# 所有處理檔案的類別都在這裡，記得引用：

```
using System;
using System.IO; // 必加！
```

三大常用類別

1. **File**: 對「檔案」的操作 (建立、刪除、檢查存在)。
2. **Directory**: 對「資料夾」的操作。
3. **Path**: 對「路徑字串」的處理 (合併路徑、取附檔名)。

FILE 類別常用方法

```
string path = "data.txt";

// 1. 檢查檔案是否存在
if (File.Exists(path)) {
    Console.WriteLine("檔案存在！");
}

// 2. 快速寫入（覆蓋）
File.WriteAllText(path, "Hello World");

// 3. 快速寫入（追加）
File.AppendAllText(path, "\nNew Line");

// 4. 快速讀取全部
string content = File.ReadAllText(path);
```

> **注意：** `WriteAllText` 適合小檔案，大檔案請用串流 (Stream)。

DIRECTORY 與 PATH

```
// 建立資料夾
Directory.CreateDirectory("MyFolder");

// 檢查資料夾
if (Directory.Exists("MyFolder")) { ... }

// 合併路徑（最強大！）
// 不要自己寫 "Folder" + "\\" + "file.txt"
// 因為 Windows 用 \, Mac/Linux 用 /
string fullPath = Path.Combine("MyData", "2024", "log.txt");
// 結果：MyData\2024\log.txt (Windows)
```

串流讀寫 (STREAM)

對於更細緻的操作，我們使用 `StreamWriter` 和 `StreamReader`。

寫入檔案 (WRITER)

```
// true 代表 append (追加模式)，false 代表覆蓋
using (StreamWriter sw = new StreamWriter("log.txt", true))
{
    sw.WriteLine($"[Log] 程式啟動於 {DateTime.Now}");
    sw.WriteLine("系統正常執行中...");
}
// using 區塊結束後，檔案會自動關閉 (Close/Dispose)
```

讀取檔案 (READER)

```
if (File.Exists("log.txt"))
{
    using (StreamReader sr = new StreamReader("log.txt"))
    {
        string line;
        // 一行一行讀，直到讀不到為止 (null)
        while ((line = sr.ReadLine()) != null)
        {
            Console.WriteLine(line);
        }
    }
}
```

實作：極簡日記系統

```
string path = "diary.txt";

Console.WriteLine("請輸入今天的日記 (輸入 'exit' 離開)：");

while (true)
{
    string input = Console.ReadLine();
    if (input == "exit") break;

    string log = $"{DateTime.Now:yyyy-MM-dd HH:mm} | {input}";

    // 寫入檔案
    File.AppendAllText(path, log + Environment.NewLine);
    Console.WriteLine("已儲存！");
}

// 顯示目前內容
Console.WriteLine("\n--- 日記內容 ---");
Console.WriteLine(File.ReadAllText(path));
```

絕對路徑 VS 相對路徑

- **相對路徑 (Relative Path)：** `"data.txt"`
  - 相對於「執行檔 (.exe)」的位置。
  - 推薦使用，搬移程式時不會壞掉。
- **絕對路徑 (Absolute Path)：** `"C:\\Users\\Admin\\Desktop\\data.txt"`
  - 寫死了完整位置。
  - **盡量避免**，除非別人的電腦也有這個路徑。

總結

1. 引用 `using System.IO;`
2. 使用 `File.Exists()` 檢查檔案。
3. 小檔案用 `File.WriteAllText` / `File.ReadAllText` 。
4. 複雜讀寫用 `StreamWriter` / `StreamReader` 搭配 `using` 。
   5.路徑處理交給 `Path.Combine()` 。

進階：CSV 檔案讀取

CSV (Comma-Separated Values) 是最常見的資料交換格式。

例如：`Name,Score,Age`

```
string[] lines = File.ReadAllLines("data.csv");

// 跳過第一行標題 (i=1)
for (int i = 1; i < lines.Length; i++)
{
    // 用逗號切割
    string[] parts = lines[i].Split(',');

    string name = parts[0];
    int score = int.Parse(parts[1]);

    Console.WriteLine($"{name} 得分 {score}");
}
```

這在處理 Excel 匯出的資料時非常有用！

未來展望：C# 的無限可能

C# 是一個功能極其強大的語言，這學期我們只學了皮毛。

1. 遊戲開發 (UNITY)

這是各位目前的主線任務！

2. 網站後端 (ASP.NET CORE)

用 C# 寫伺服器 API，處理資料庫連線、會員系統。是目前業界薪資非常高的領域。

3. 視窗程式 (WPF / WINDOWS FORMS)

開發 Windows 桌面軟體，例如 POS 系統、工廠管理系統。

保持好奇心，持續學習！

下一章：課程總結與 UNITY 銜接