Лабораторна робота №13
Робота з файловою системою. Перегляд та навігація

Методичні вказівки

Більшість завдань у програмуванні так чи інакше пов'язані з роботою з файлами та каталогами. Нам може знадобитися прочитати текст з файлу або зробити запис, видалити файл або цілий каталог, не кажучи вже про більш комплексні завдання, як, наприклад, створення текстового редактора та інші подібні завдання.

Фреймворк .NET надає великі можливості з управління та маніпуляції файлами та каталогами, які здебільшого зосереджені у просторі імен System.IO. Класи, розташовані в цьому просторі імен (такі як Stream, StreamWriter, FileStream та ін), дозволяють керувати файловим введенням-виводом.

Робота з дисками

Роботу з файловою системою почнемо з найвищого рівня - дисків. Для представлення диска у просторі імен System.IO є клас DriveInfo.

Цей клас має статичний метод GetDrives(), який повертає імена логічних дисків комп'ютера. Також він надає низку корисних властивостей:

AvailableFreeSpace: вказує на обсяг доступного вільного місця на диску в байтах

DriveFormat: отримує ім'я файлової системи

DriveType: представляє тип диска

IsReady: чи готовий диск (наприклад, DVD-диск може бути не вставлений у дисковод)

Name: отримує ім'я диска

RootDirectory: повертає кореневий каталог диска

TotalFreeSpace: отримує загальний обсяг вільного місця на диску в байтах

TotalSize: загальний розмір диска в байтах

VolumeLabel: отримує або встановлює мітку тома

Отримаємо імена та властивості всіх дисків на комп'ютері:

```
DriveInfo[] drives = DriveInfo.GetDrives();

foreach (DriveInfo drive in drives)
{
    Console.WriteLine($"Назва: {drive.Name}");
    Console.WriteLine($"Тип: {drive.DriveType}");
    if (drive.IsReady)
    {
        Console.WriteLine($"Об’єм диску: {drive.TotalSize}");
        Console.WriteLine($"Вільний простір: {drive.TotalFreeSpace}");
```

```
            Console.WriteLine($"Мітка диску: {drive.VolumeLabel}");
        }
        Console.WriteLine();
    }
```

Робота з каталогами

Клас Directory

Статичний клас Directory надає ряд методів управління каталогами. Деякі з цих методів:

CreateDirectory(path): створює каталог по вказаному шляху path

Delete(path): видаляє каталог по вказаному шляху path

Exists(path): визначає, чи існує каталог на вказаному шляху path. Якщо існує, повертається true, якщо не існує, false

GetCurrentDirectory(): отримує шлях до поточної папки

GetDirectories(path): отримує список підкаталогів у каталозі path

GetFiles(path): отримує список файлів у каталозі path

GetFileSystemEntries(path): отримує список підкаталогів та файлів у каталозі path

Move(sourceDirName, destDirName): переміщує каталог

GetParent(path): отримання батьківського каталогу

GetLastWriteTime(path): повертає час останньої зміни каталогу

GetLastAccessTime(path): повертає час останнього звернення до каталогу

GetCreationTime(path): повертає час створення каталогу

Клас DirectoryInfo

Цей клас надає функціональність для створення, видалення, переміщення та інших операцій з каталогами. Багато в чому він схожий на Directory, але не статичним.

Для створення об'єкта класу DirectoryInfo застосовується конструктор, який як параметр приймає шлях до каталогу:

```
public DirectoryInfo (string path);
```

Основні методи класу DirectoryInfo:

Create(): створює каталог

CreateSubdirectory(path): створює підкаталог по вказаному шляху path

Delete(): видаляє каталог

GetDirectories(): отримує список підкаталогів папки у вигляді масиву DirectoryInfo

GetFiles(): отримує список файлів у папці у вигляді масиву FileInfo

MoveTo(destDirName): переміщує каталог

Основні властивості класу DirectoryInfo:

CreationTime: представляє час створення каталогу
LastAccessTime: представляє час останнього доступу до каталогу
LastWriteTime: представляє час останньої зміни каталогу
Exists: визначає, чи існує каталог
Parent: отримання батьківського каталогу
Root: отримання кореневого каталогу
Name: ім'я каталогу
FullName: повний шлях до каталогу

Directory або DirectoryInfo

Як видно з функціоналу, обидва класи надають схожі можливості. Коли і що використовувати? Якщо потрібно здійснити одну-дві операції з одним каталогом, то простіше використати клас Directory. Якщо необхідно виконати послідовність операцій з одним і тим самим каталогом, краще скористатися класом DirectoryInfo. Чому? Справа в тому, що методи класу Directory виконують додаткові перевірки безпеки. А для класу DirectoryInfo такі перевірки не завжди є обов'язковими.

Подивимося на прикладах застосування цих класів.

Отримання списку файлів та каталогів

```
string dirName = "C:\\";
// якщо каталог існує
if (Directory.Exists(dirName))
{
    Console.WriteLine("Підкаталоги:");
    string[] dirs = Directory.GetDirectories(dirName);
    foreach (string s in dirs)
    {
        Console.WriteLine(s);
    }
    Console.WriteLine();
    Console.WriteLine("Файли:");
    string[] files = Directory.GetFiles(dirName);
    foreach (string s in files)
    {
        Console.WriteLine(s);
    }
}
```

Фільтрація каталогів та файлів

Знайдемо всі папки, які починаються на “books”

```
// класс Directory
string[] dirs = Directory.GetDirectories(dirName, "books*.");

// класс DirectoryInfo
var directory = new DirectoryInfo(dirName);
```

```
DirectoryInfo[] dirs = directory.GetDirectories("books*.");
```

Або отримаємо всі файли з розширенням “*. exe”:

```
// клас Directory
string[] files = Directory.GetFiles(dirName, "*.exe");

// клас DirectoryInfo
var directory = new DirectoryInfo(dirName);
FileInfo[] files = directory.GetFiles("*.exe");
```

Отримання інформації про каталог

```
string dirName = "C:\\Program Files";

DirectoryInfo dirInfo = new DirectoryInfo(dirName);

Console.WriteLine($"Назва каталога: {dirInfo.Name}");
Console.WriteLine($"Повна назва каталога: {dirInfo.FullName}");
Console.WriteLine($"Час створення каталога: {dirInfo.CreationTime}");
Console.WriteLine($"Корневий каталог: {dirInfo.Root}");
```

Работа с файлами. Классы File и FileInfo

Подібно до пари Directory/DirectoryInfo для роботи з файлами призначена пара класів File і FileInfo. З їхньою допомогою ми можемо створювати, видаляти, переміщати файли, отримувати їх властивості та багато іншого.

Деякі корисні методи та властивості класу FileInfo:

CopyTo(path): копіює файл у нове місце за вказаним шляхом path

Create(): створює файл

Delete(): видаляє файл

MoveTo(destFileName): переміщує файл у нове місце

Властивість Directory: отримує батьківський каталог як об'єкт DirectoryInfo

Властивість DirectoryName: отримує повний шлях до батьківського каталогу

Властивість **Exists**: вказує, чи є файл

Властивість **Length**: отримує розмір файлу

Властивість **Extension**: отримує розширення файлу

Властивість **Name**: отримує ім'я файлу

Властивість **FullName**: отримує повне ім'я файлу

Клас File реалізує схожу функціональність за допомогою статичних методів:

Copy(): копіює файл у нове місце

Create(): створює файл

Delete(): видаляє файл

Move: переміщує файл у нове місце

Exists(file): визначає, чи існує файл

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
    Console.WriteLine("Им’я файла: {0}", fileInf.Name);
    Console.WriteLine("Час створення: {0}", fileInf.CreationTime);
    Console.WriteLine("Розмір: {0}", fileInf.Length);
}
```

Індивідуальне завдання:

1.Розробити Windows Form (або WPF) додаток, який виконуватиме наступні функції:

- завантаження списку дисків при старті програми;
- переміщення по файловій системи
- перегляд основних властивостей виділеного диску;
- перегляд основних властивостей виділеного каталогу;
- перегляд основних властивостей виділеного файлу;
- фільтрацію списку файлів;
- фільтрацію списку каталогів.

2. Додаткової функції:

- перегляд вмісту графічних файлів;
- перегляд атрибутів безпеки файлів та каталогів;
- перегляд вмісту текстових файлів.

3. Дизайн користувацького інтерфейсу, вибір елементів керування на вибір студента.