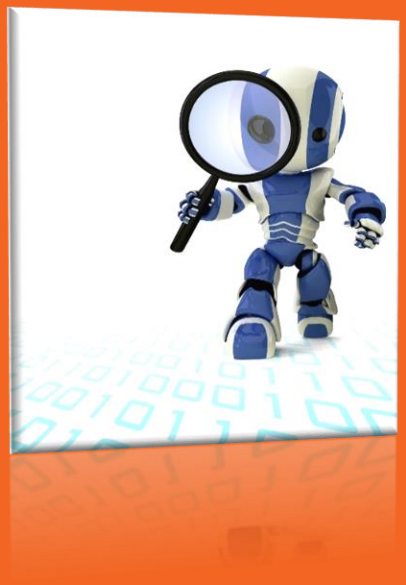
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 2

BÀI 3: THAO TÁC VỚI TẬP TIN VÀ THƯ MỤC

www.poly.edu.vn

Mục tiêu

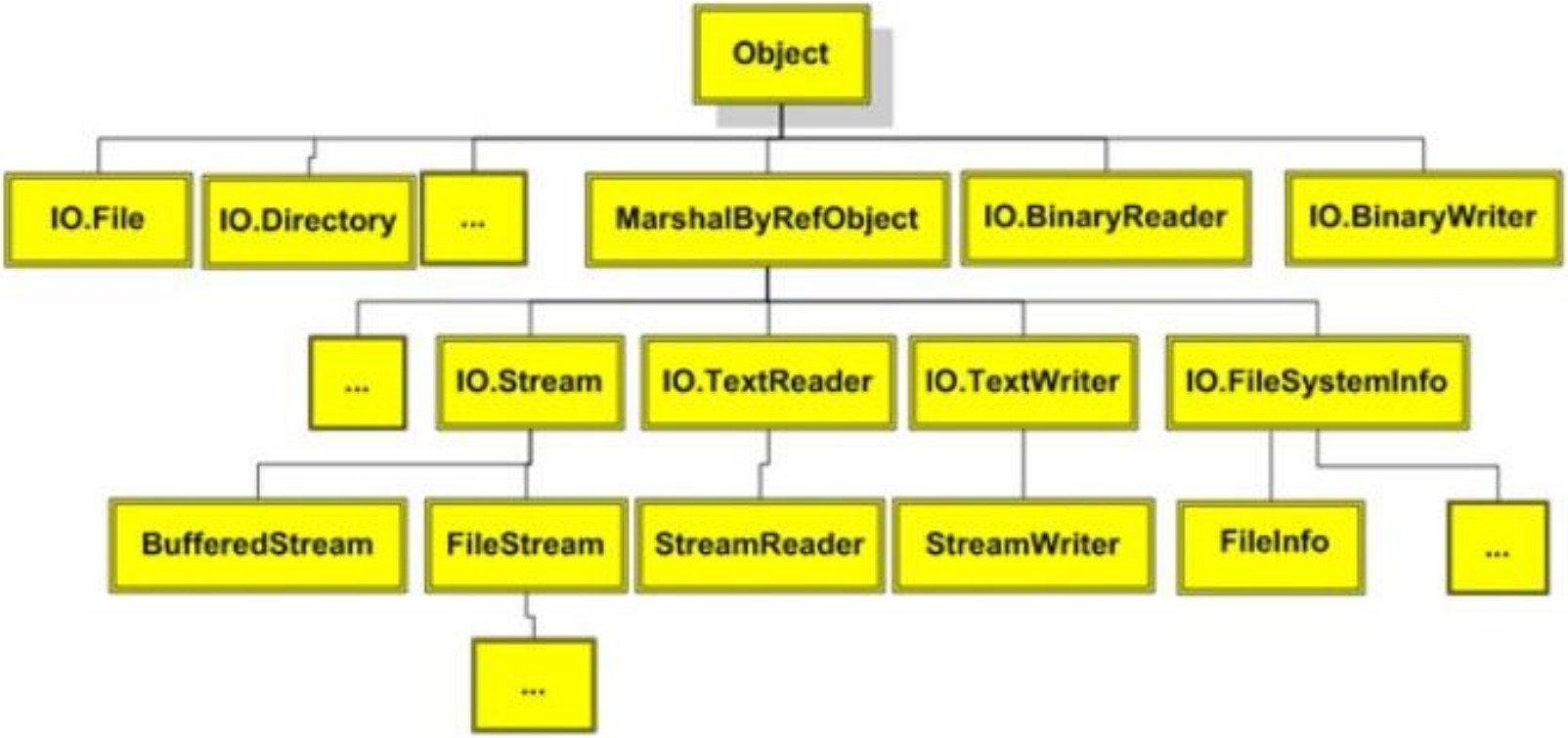
- C# System.IO Namespace
- Class File và Directory

System.IO Namespace

- System.IO Namespace có các lớp khác nhau được sử dụng để thực hiện nhiều hoạt động với các tập tin, chẳng hạn như việc tạo và xóa các tập tin, đọc hoặc viết vào một tập tin, đóng một tập tin.
- Một tập tin là một tập hợp các dữ liệu được lưu trữ trong một đĩa với một tên cụ thể và một đường dẫn thư mục. Khi một tập tin được mở để đọc hoặc viết, nó sẽ trở thành một luồng tin.

System.IO Namespace

- Một số lớp thường sử dụng trong System.IO



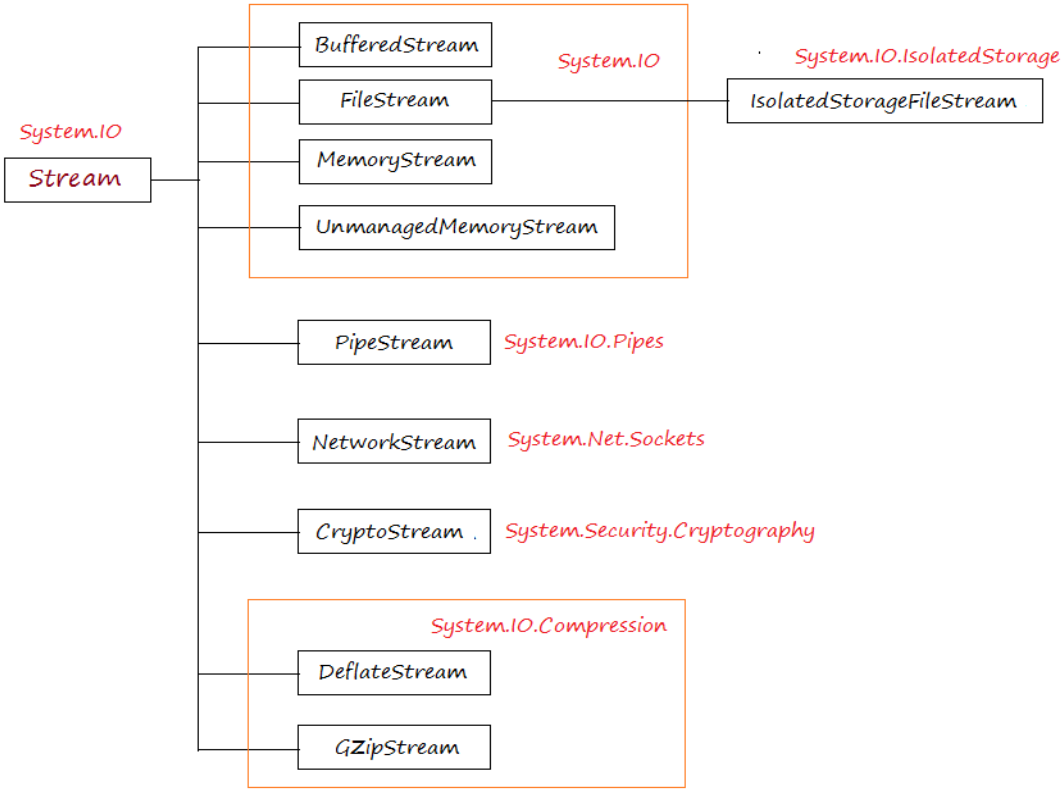
System.IO Namespace

- Một số lớp thường sử dụng trong System.IO

Lớp I/O	Mô tả
BinaryReader	Đọc dữ liệu nguyên bản từ một dòng nhị phân.
BinaryWriter	Đọc dữ liệu nguyên bản từ một dòng nhị phân.
BufferedStream	Một lưu trữ tạm thời cho một dòng các byte.
Directory	Giúp trong việc điều khiển một cấu trúc thư mục.
DirectoryInfo	Được sử dụng để thực hiện các hoạt động trên các thư mục.
DriveInfo	Cung cấp thông tin cho các ổ đĩa.
File	Giúp thao tác tập tin.
FileInfo	Được sử dụng để thực hiện các hoạt động trên các tập tin.
FileStream	Được sử dụng để đọc và ghi vào vị trí bất kỳ trong một tập tin.
MemoryStream	Được sử dụng để truy cập ngẫu nhiên luồng dữ liệu lưu trong bộ nhớ.
Path	Thực hiện các hoạt động về thông tin đường dẫn.
StreamReader	Được sử dụng để đọc các ký tự từ một dòng byte.
StreamWriter	Được sử dụng để ghi các ký tự lên một dòng byte.
StringReader	Được sử dụng để đọc từ một bộ đệm chuỗi.
StringWriter	Được sử dụng để ghi lên một bộ đệm chuỗi.

FileStream

- Các lớp FileStream trong namespace System.IO giúp đọc, ghi và đóng file. Lớp này xuất phát từ lớp trừu tượng Stream.
- Mối liên hệ thừa kế các class trong System.IO



- Bạn cần phải tạo ra một đối tượng FileStream để tạo ra một tập tin mới hoặc mở một tập tin hiện có. Cú pháp để tạo một đối tượng FileStream là như sau:

```
FileStream <Ten-doi-tuong> = new FileStream( <ten_file>, <FileMode Enumerator>,
                                             <FileAccess Enumerator>, <FileShare Enumerator>);
```

- Ví dụ, chúng ta tạo ra một đối tượng FileStream F cho đọc một tập tin có tên sample.text như sau:

```
FileStream F = new FileStream("sample.txt", FileMode.Open, FileAccess.Read, FileShare.Read);
```

Các tham số trong FileStream

Tham số	Mô tả
FileMode	**FileMode** chứa các phương thức khác nhau để mở tập tin. Các thành viên cảu FileMode là: • **Append**: Nó mở ra một tập tin hiện có và đặt con trỏ ở cuối của tập tin, hoặc tạo ra các tập tin, nếu các tập tin không tồn tại. • **Create**: Nó tạo ra một tập tin mới. • **CreateNew**: Nó chỉ định cho hệ điều hành, là nó sẽ tạo ra một tập tin mới. • **Open**: Nó mở ra một tập tin hiện có. • **OpenOrCreate**: Nó chỉ định cho hệ điều hành là nó sẽ mở ra một tập tin nếu nó tồn tại, nếu không nó sẽ tạo ra một tập tin mới. • **Truncate**: Nó mở ra một tập tin hiện có và đặt kích thước về 0 byte.
FileAccess	**FileAccess** bao gồm các thành phần: **Read**, **ReadWrite** and **Write**.
FileShare	**FileShare** bao gồm các thành viên sau: • **Inheritable**: Nó cho phép một xử lý tập tin cho phép thừa kế bởi các tiến trình con • **None**: Nó từ chối chia sẻ các tập tin hiện hành • **Read**: Nó cho phép mở tập tin để đọc • **ReadWrite**: Nó cho phép mở tập tin để đọc và viết • **Write**: Nó cho phép mở tập tin để ghi

Ghi file với FileStream

```
static void Main(string[] args)
{
    string fpath = @"D:\Test.txt";
    // Delete file if exists
    if (File.Exists(fpath))
    {
        File.Delete(fpath);
    }
    // Create the file
    using (FileStream fs = File.Create(fpath))
    {
        AddTexttoFile(fs, "Hi");
        AddTexttoFile(fs, "\r\nWelcome to FPoly");
        AddTexttoFile(fs, "\r\nFileStream Example");
    }
}
private static void AddTexttoFile(FileStream fs, string value)
{
    byte[] info = new UTF8Encoding(true).GetBytes(value);
    fs.Write(info, 0, info.Length);
}
```

Đọc file với FileStream

```
class Program
{
    static void Main(string[] args)
    {
        string fpath = @"D:\Test.txt";
        // Check if file exists
        if (File.Exists(fpath))
        {
            // Open the file and read
            using (FileStream fs = File.OpenRead(fpath))
            {
                byte[] b = new byte[1024];
                UTF8Encoding encode = new UTF8Encoding(true);
                while (fs.Read(b, 0, b.Length) > 0)
                {
                    Console.WriteLine(encode.GetString(b));
                }
            }
        }
        Console.ReadLine();
    }
}
```

- TextWriter là lớp cơ sở trừu tượng của StreamWriter và StringWriter, dùng ghi một chuỗi ký tự hoặc text

```
static void Main(string[] args)
{
    string fpath = @"D:\Test.txt";
    // Check file if exists
    if (File.Exists(fpath))
    {
        File.Delete(fpath);
    }
    // Create the file
    using (TextWriter wr = File.CreateText(fpath))
    {
        wr.WriteLine("Hi");
        wr.WriteLine("\r\nWelcome to FPoly");
        wr.WriteLine("\r\nTextWriter Example");
    }
}
```

TEXTREADER

- TextReader là lớp trừu tượng cung cấp các phương thức có khả năng đọc một chuỗi ký tự hoặc text.
- Không thể tạo đối tượng trực tiếp từ lớp TextReader
- Sau khi đọc hoặc ghi dữ liệu đều cần phải giải phóng vùng nhớ trực tiếp hoặc gián tiếp:
  - Trực tiếp dùng phương thức Dispose trong khối try/catch
  - Gián tiếp dùng khối "using"

TEXTREADER

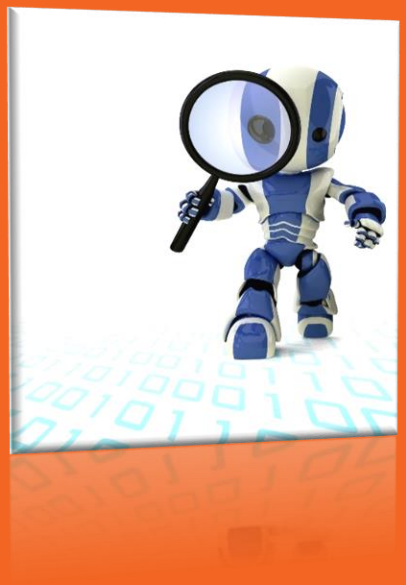
```
class Program
{
    static void Main(string[] args)
    {
        string filepath = @"D:\csharpfile.txt";

        //Read One Line
        using(TextReader tr=File.OpenText(filepath))
        {
            Console.WriteLine(tr.ReadLine());
        }

        //Read 4 Characters
        using (TextReader tr = File.OpenText(filepath))
        {
            char[] ch = new char[4];
            tr.ReadBlock(ch, 0, 4);
            Console.WriteLine(ch);
        }

        //Read full file
        using (TextReader tr = File.OpenText(filepath))
        {
            Console.WriteLine(tr.ReadToEnd());
        }
        Console.ReadKey();
    }
}
```

DEMO

- Hiện thực các ví dụ

FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 2

BÀI 3: THAO TÁC VỚI TẬP TIN VÀ THƯ MỤC (P2)

www.poly.edu.vn

BINARYWRITER

- Hỗ trợ ghi dữ liệu thuộc các kiểu nguyên thủy như int, unint, char…và cả kiểu chuỗi string

```
class Program
{
    static void Main(string[] args)
    {
        string fpath = @"D:\Test.txt";
        // Check file if exists
        if (File.Exists(fpath))
        {
            File.Delete(fpath);
        }
        using (BinaryWriter bw = new BinaryWriter(File.Open(fpath,FileMode.Create)))
        {
            bw.Write(1.25);
            bw.Write("Welcome to FPoly");
            bw.Write(10);
            bw.Write(true);
            bw.Write("test");
        }
    }
}
```

Đọc các file nhị phân (.bin)

```
        static void Main(string[] args)
        {
            string fpath = @"D:\Test.txt";
            // Writing values to file
            if (File.Exists(fpath))
            {
                File.Delete(fpath);
            }
            using (BinaryWriter bw = new BinaryWriter(File.Open(fpath, FileMode.Creat
e)))
            {
                bw.Write(1.25);
                bw.Write("Welcome to FPoly");
                bw.Write(10);
                bw.Write(true);
                bw.Write("test");
            }
            // Reading Values by creating BinaryReader instance
            using (BinaryReader br = new BinaryReader(File.Open(fpath, FileMode.Open)))
            {
                Console.WriteLine(br.ReadDouble());
                Console.WriteLine(br.ReadString());
                Console.WriteLine(br.ReadInt32());
                Console.WriteLine(br.ReadBoolean());
                Console.WriteLine(br.ReadString());
            }
            Console.ReadLine();
        }
    }
```

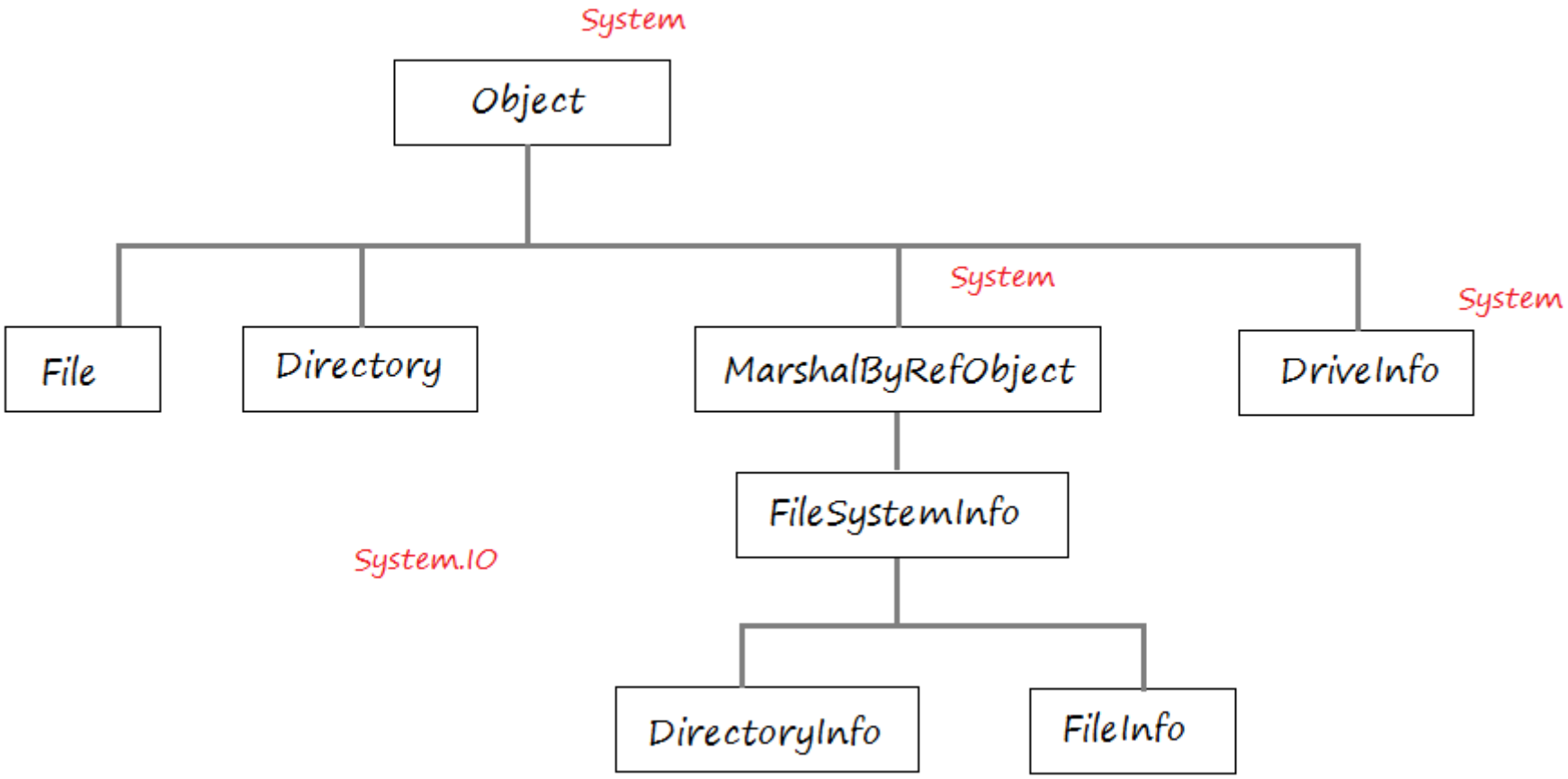
String Writer và StringReader

- Sử dụng StringReader và StringWriter lưu trữ và đọc dữ liệu chuỗi

```
static void Main(string[] args)
{
    string text = "Hello. This is Line 1 \n Hello. This is Line 2 \nHello This is Line 3";
    //Writing string into StringBuilder
    StringBuilder sb = new StringBuilder();
    StringWriter writer = new StringWriter(sb);
    //Store Data on StringBuilder
    writer.WriteLine(text);
    writer.Flush();
    writer.Close();

    //Read Entry
    StringReader reader = new StringReader(sb.ToString());
    //Check to End of File
    while (reader.Peek() > -1)
    {
        Console.WriteLine(reader.ReadLine());
    }
}
```

CLASS FILE VÀ DIRECTORY

- Sơ đồ quan hệ thừa kế giữa các class



CLASS FILE VÀ DIRECTORY

- Lớp Directory có nhiều phương thức dành cho việc tạo, di chuyển, duyệt thư mục
- Các phương thức trong lớp Directory đều là method static
- Lớp DirectoryInfo cũng cung cấp những phương thức mà lớp Directory có, đồng thời bổ sung thêm nhiều phương thức hữu ích hơn cho việc duyệt cấu trúc cây thư mục
- Lớp *DirectoryInfo* không có phương thức *static*, vì vậy muốn sử dụng các method cần tạo object với tham số cần truyền vào cho *DirectoryInfo* là path của folder.

CLASS FILE VÀ DIRECTORY

- Một số phương thức quan trọng của class Directory

Tên phương thức	Mô tả
CreateDirectory()	Tạo tất cả các thư mục và thư mục con trong đường dẫn tham số.
Delete()	Xóa thư mục và các nội dung của nó.
Exists()	Trả về kiểu bool (*true*: tồn tại, *false*: không tồn tại)
GetCreationTime()	Get / Set ngày giờ tạo thư mục (作成時間)
SetCreationTime()	
GetCurrentDirectory()	Get / Set thư mục hiện hành.
SetCurrentDirectory()	
GetDirectories()	Trả về một array các folder con của một folder.
GetDirectoryRoot()	Trả về folder gốc của đường dẫn.
GetFiles()	Trả về một *array* tên các files có trong folder.
GetLastAccessTime()	Get / Set ngày giờ của lần truy cập cuối cùng đến folder.
SetLastAccessTime()	

- Một số Method / Property của class DirectoryInfo

Method / Property	Ý nghĩa
Attributes	Get / Set thuộc tính của file hoặc directory hiện tại.
CreationTime	Get / Set thời gian tạo file hoặc directory hiện tại.
Exists	Trả về *true*: tồn tại, *false*: không tồn tại.
Extension	Lấy chuỗi đại diện cho phần mở rộng của file.
FullName	Lấy đường dẫn đầy đủ của file hoặc directory.
LastAccessTime	Get / Set thời gian file hoặc directory hiện tại được truy cập lần cuối.
LastWriteTime	Get / Set thời gian khi file hoặc directory hiện tại được ghi lần cuối.
Name	Lấy tên thư mục (tên instace của *DirectoryInfo*)
Parent	Lấy parent directory của một subdirectory cụ thể.
Root	Lấy phần root của directory.
Create()	Tạo directory.
CreateSubdirectory()	Tạo subdirectory hoặc subdirectories cho path đã chỉ định. Đường dẫn đã chỉ định có thể liên quan đến instance của lớp *DirectoryInfo*.
Delete()	Xóa System.IO.DirectoryInfo này nếu nó là empty.
GetDirectories()	Trả về các thư mục con của thư mục hiện tại.

Ví dụ tạo directory

```
static void Main(string[] args)
{
    //Specify the directory which we want to manipulate
    string fpath = @"D:\Test";
    DirectoryInfo di = new DirectoryInfo(fpath);
    // Check if directory exists
    if (di.Exists)
    {
        Console.WriteLine("Directory Already Exists");
    }
    else
    {
        // Create directory
        di.Create();
        Console.WriteLine("Directory Created Successfully");
    }
    Console.ReadLine();
}
```

Ví dụ copy directory

```
static void Main(string[] args)
{
    //Specify the directory which we want to manipulate
    string sourcedir = @"D:\FPoly";
    string targetdir = @"D:\Test";
    DirectoryInfo sdi = new DirectoryInfo(sourcedir);
    DirectoryInfo tdi = new DirectoryInfo(targetdir);
    // Check if target directory exists, if not, create it
    if (!tdi.Exists)
    {
        tdi.Create();
    }
    // Copy each file into it's new directory.
    foreach (FileInfo fi in sdi.GetFiles())
    {
        fi.CopyTo(Path.Combine(tdi.ToString(), fi.Name), true);
        Console.WriteLine(@"Copying {0}\{1}", tdi.FullName, fi.Name);
    }

    // Copy each subdirectory and it's files
    foreach (DirectoryInfo sourceSubDir in sdi.GetDirectories())
    {
        DirectoryInfo targetSubDir =
            tdi.CreateSubdirectory(sourceSubDir.Name);
        // Copy each file into it's new directory.
        foreach (FileInfo fi in sourceSubDir.GetFiles())
        {
            fi.CopyTo(Path.Combine(targetSubDir.ToString(), fi.Name), true);
            Console.WriteLine(@"Copying {0}\{1}", targetSubDir.FullName, fi.Name);
        }
    }
```

Một số phương thức quan trọng của class File

Tên phương thức	Mô tả
AppendText()	Tạo một *StreamWriter* nối thêm văn bản được mã hóa UTF-8 vào một file hiện có hoặc vào một file mới nếu file được chỉ định không tồn tại.
AppendAllText()	Mở một file, nối thêm chuỗi đã chỉ định vào file và sau đó đóng file. Nếu file không tồn tại, phương thức này tạo một file, ghi chuỗi đã chỉ định vào file đó, sau đó đóng file.
AppendAllLines()	Nối các dòng vào một file, rồi đóng file. Nếu file đã chỉ định không tồn tại, phương thức này sẽ tạo một file, ghi các dòng được chỉ định vào file đó và sau đó đóng file.
Copy()	Sao chép file hiện có vào một file mới. Ghi đè một tập tin cùng tên không được phép.
Create()	Tạo hoặc ghi đè file trong đường dẫn đã chỉ định.
CreateText()	Tạo hoặc mở file để ghi dữ liệu với UTF-8 encoded.
Delete()	Xóa file được chỉ định.
Exists()	Trả về *true*: tồn tại, *false*: không tồn tại.
GetAttributes()	Get / Set *FileAttribution* của file trên đường dẫn.
SetAttributes()	
GetCreationTime()	Get / Set ngày và thời gian tạo của file hoặc directory chỉ định.
SetCreationTime()	

Một số Method / Property của class FileInfo

Method / Property	Ý nghĩa
Attributes()	Get / Set thuộc tính của file hoặc directory hiện tại.
CreationTime	Get / Set thời gian tạo của file hoặc directory hiện tại.
Directory	Lấy một instance của parent directory.
Exists	Trả về *true*: tồn tại, *false*: không tồn tại.
Extension	Lấy chuỗi đại diện cho phần mở rộng của file.
FullName	Lấy đường dẫn đầy đủ của file hoặc directory.
LastAccessTime	Get / Set thời gian file hoặc directory hiện tại được truy cập lần cuối.
LastWriteTime	Get / Set thời gian khi file hoặc directory hiện tại được ghi lần cuối.
Length	Lấy kích thước tính bằng byte của file hiện tại.
Name	Lấy tên file.
AppendText()	Tạo một *StreamWriter* để thêm text (appends text) vào file được đại diện bởi instance của *FileInfo*.
CopyTo()	Sao chép một file hiện có sang một file mới, không cho phép ghi đè lên file hiện có.
Create()	Tạo file.
Delete()	Xóa file vĩnh viễn.

Ví dụ tạo file

```
static void Main(string[] args)
{
    string fpath = @"D:\Test.txt";
    // Check file if exists
    if (File.Exists(fpath))
    {
        File.Delete(fpath);
    }
    // Create the file
    FileInfo fi = new FileInfo(fpath);
    //fi.Create();
    // Create and write data to file
    using (StreamWriter sw = fi.CreateText())
    {
        sw.WriteLine("Hi");
        sw.WriteLine("\r\nWelcome to FPoly");
        sw.WriteLine("\r\nFileInfo Example");
    }
}
```

Ví dụ đọc file

```
static void Main(string[] args)
{
    string fpath = @"D:\Test.txt";
    // Check if file exists
    if (File.Exists(fpath))
    {
        FileInfo fi = new FileInfo(fpath);
        // open the file to read text
        using (StreamReader sr = fi.OpenText())
        {
            string txt;
            // Read the data from file, until the end of file is reache

            while ((txt = sr.ReadLine()) != null)
            {
                Console.WriteLine(txt);
            }
        }
    }
    Console.ReadLine();
}
```

DEMO

- Hiện thực các ví dụ

Tổng kết bài học

- C# System.IO Namespace
- Class File và Directory

KẾT THÚC