파일 다루기

이것이 C#이다

Contents

- 파일 정보와 디렉토리 정보 다루기
- 파일을 읽고 쓰기 위해 알아야 할 것들
- 18.3 이진 데이터 처리, BinaryWriter/BinaryReader
- 텍스트 파일 처리, StreamWriter/StreamReader
- 객체 직렬화하기

18.1 파일 정보와 디렉토리 정보 다루기(1)

❖ System.IO 네임스페이스의 클래스

클래스	설명
File	파일의 생성, 복사, 삭제, 이동, 조회를 처리하는 정적 메소드를 제공합니다.
FileInfo	File 클래스와 하는 일은 동일하지만 정적 메소드 대신 인스턴스 메소드를 제공합니다.
Directory	디렉토리의 생성, 삭제, 이동, 조회를 처리하는 정적 메소드를 제공합니다.
DirectoryInfo	Directory 클래스와 하는 일은 동일하지만 정적 메소드 대신 인스턴스 메소드를 제공합니다.

❖ 주요 메소드와 프로퍼티

기능	File	FileInfo	Directory	DirectoryInfo
생성	Create( )	Create( )	CreateDirectory( )	Create( )
복사	Copy( )	CopyTo( )	–	–
삭제	Delete( )	Delete( )	Delete( )	Delete( )
이동	Move( )	MoveTo( )	Move( )	MoveTo( )
존재 여부 확인	Exists( )	Exists	Exists( )	Exists
속성 조회	GetAttributes( )	Attributes	GetAttributes( )	Attributes
하위 디렉토리 조회	–	–	GetDirectories( )	GetDirectories( )
하위 파일 조회	–	–	GetFiles( )	GetFiles( )

18.1 파일 정보와 디렉토리 정보 다루기(2)

- File 클래스와 FileInfo 클래스의 사용 스타일
- Directory 클래스와 DirectoryInfo 클래스의 사용 방법

기능	Directory	DirectoryInfo
생성	DirectoryInfo dir = Directory.CreateDirectory("a");	DirectoryInfo dir = new DirectoryInfo("a"); dir.Create( );
삭제	Directory.Delete("a");	DirectoryInfo dir = new DirectoryInfo("a"); dir.Delete( );
이동	Directory.Move("a", "b");	DirectoryInfo dir = new DirectoryInfo("a"); dir.MoveTo("b");
존재 여부 확인	if (Directory.Exists("a.dat")) // …	DirectoryInfo dir = new DirectoryInfo("a"); if (dir.Exists) // …
속성 조회	Console.WriteLine(Directory.GetAttributes("a"));	DirectoryInfo dir = new DirectoryInfo("a"); Console.WriteLine(dir.Attributes);
하위 디렉토리 조회	string[ ] dirs = Directory.GetDirectories("a");	DirectoryInfo dir = new DirectoryInfo("a"); DirectoryInfo[ ] dirs = dir.GetDirectories( );
하위 파일 조회	string[ ] files = Directory.GetFiles("a");	DirectoryInfo dir = new DirectoryInfo("a"); FileInfo[ ] files = dir.GetFiles( );

18.1 파일 정보와 디렉토리 정보 다루기(3)

❖ 18.1.1 예제 프로그램: 디렉토리/파일 정보 조회하기

- 매개 변수를 입력하지 않으면 현재 디렉토리 조회
- 매개 변수 입력하면, 입력한 디렉터리 조회
- 출력
  - 디렉터리: 이름, 속성
  - 파일:이름, 크기, 속성
- 데모 예제 - Dir

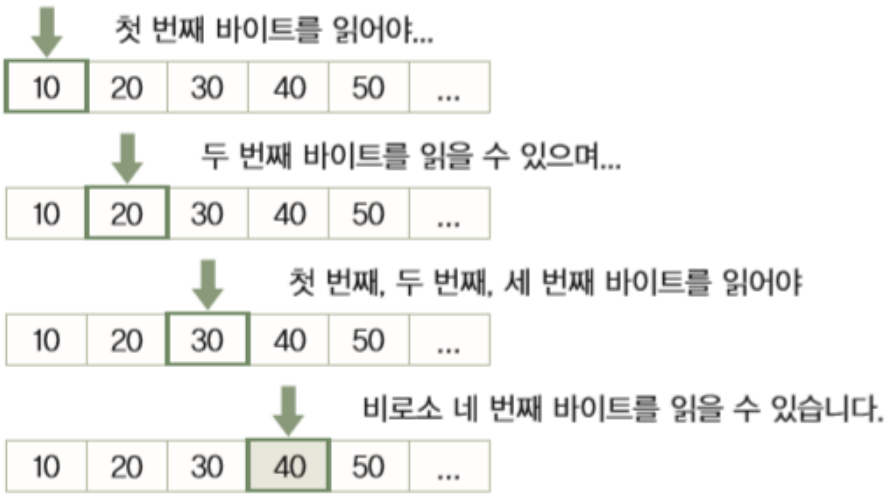
❖ 18.1.2 예제 프로그램: 디렉토리/파일 생성하기

- 매개 변수로 입력 받은 경로에 새 디렉터리나 파일 생성
- 이미 존재하는 경우 최종 수정시간만 갱신
- 데모 예제 - Touch

18.2 파일을 읽고 쓰기 위해 알아야 할 것들 (1)

❖ 스트림

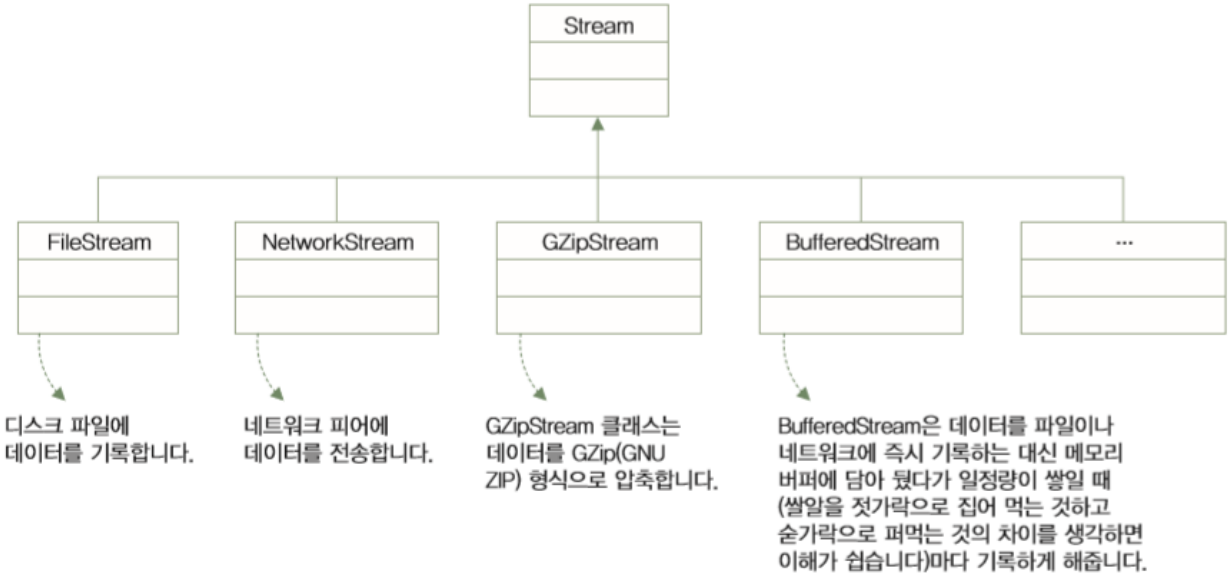
- 데이터가 흐르는 통로
- 순차 접근 방식
  - 네트워크, 데이터 백업 장치
- 임의 접근 방식
  - 하드 디스크



❖ System.IO.Stream 클래스

-
-

두 지원



18.2 파일을 읽고 쓰기 위해 알아야 할 것들 (2)

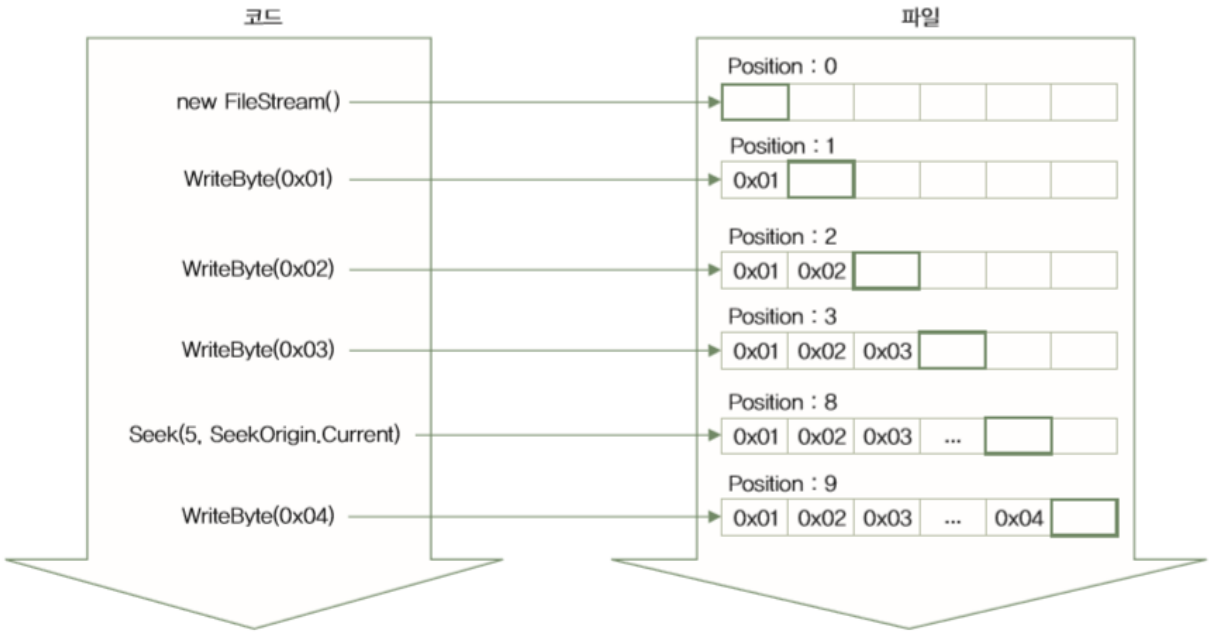
- **FileStream 클래스의 인스턴스 생성**
- **FileStream 클래스로 데이터 쓰기**
  - Stream 클래스의 Write와 WriteByte 메소드 오버라이딩
  - byte 배열로 변환해주는 BitConverter 클래스
- **FileStream 클래스로 데이터 읽어 오기**
  - Stream 클래스의 Read와 ReadByte 메소드 오버라이딩

```
byte[] rBytes = new byte[8];
// 1) 파일 스트림 생성
Stream inStream = new FileStream("a.dat", FileMode.Open);
// 2) rBytes의 길이만큼(8바이트) 데이터를 읽어 rBytes에 저장
inStream.Read(rBytes, 0, rBytes.Length);
// 3) BitConverter를 이용하여 rBytes에 담겨있는 값을 long 형식으로 변환
long readValue = BitConverter.ToInt64(rbytes, 0);
// 4) 파일 스트림 닫기
inStream.Close();
```

- **데모 예제 - BasicIO**

18.2 파일을 읽고 쓰기 위해 알아야 할 것들 (3)

❖ Write(), WriteByte(), Read(), ReadByte() 호출 이면



ɔsition 프로퍼티

1씩 자동 증가

❖ Seek()를 호출 / Position 프로퍼티에 직접 값 대입

- 임의 접근

```
Stream outStream = new FileStream("a.dat", FileMode.Create);
// …
outStream.Seek(5, SeekOrigin.Current);
outStream.WriteByte(0x04);
```

현재 위치에서 5바이트 뒤로 이동

❖ 데모 예제 - SeqNRand

18.3 이진 데이터 처리, BinaryWriter/BinaryReader

- FileStream의 byte 형식/byte의 배열 형식 변환 문제 해결
- Stream으로부터 파생된 클래스의 인스턴스와 같이 사용
- BinaryWriter를 FileStream과 함께 이용하는 예

```
BinaryWriter bw = new BinaryWriter( new FileStream("a.dat", FileMode.Create) );

bw.Write(32);
bw.Write("Good Morning!");
bw.Write(3.14);

bw.Close();
```

Write() 메소드는 C#이 제공하는 모든 기본 데이터 형식에 대해 오버로딩되어 있습니다.

- BinaryReader를 FileStream과 함께 이용하는 예

```
BinaryReader br = new BinaryReader( new FileStream("a.dat", FileMode.Open) );

int    a = br.ReadInt32();
string b = br.ReadString();
double c = br.ReadDouble ();

br.Close();
```

BinaryReader는 읽을 데이터 형식별로 ReadXXX() 메소드(XXX는 읽을 데이터 형식의 이름)를 제공합니다

- 데모 예제 - BinaryFile

18.4 텍스트 파일 처리, StreamWriter/StreamReader

- Stream으로부터 파생된 클래스의 인스턴스와 같이 사용
- StreamWriter의 사용 예제

```
StreamWriter sw = new StreamWriter( new FileStream("a.dat", FileMode.Create) );

sw.Write (32);
sw.WriteLine( "Good Morning!" );
sw.WriteLine(3.14);

sw.Close();
```

Write()와 WriteLine() 메소드는 C#이 제공하는 모든 기본 데이터 형식에 대해 오버로딩되어 있습니다.

- StreamReader의 사용 예제

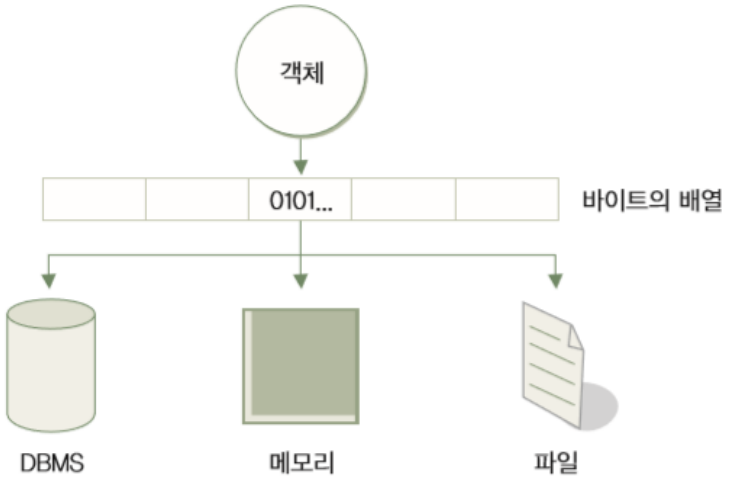
```
StreamReader sr = new StreamReader( new FileStream("a.dat", FileMode.Open) );

while ( sr.EndOfStream == false )
{
    Console.WriteLine(sr.ReadLine());
}
sr.Close();
```

EndOfStream 프로퍼티는 스트림의 끝에 도달했는지를 알려줍니다.

- 데모 예제 – TextFile

18.5 객체 직렬화하기

- **직접 정의한 클래스나 구조체 같은 복합**
- **복합 데이터 형식 쓰기와 읽기**
  - 그 형식이 가진 필드 값의 저장 순서 정의 후, 이
- **직렬화**
  - C#의 복합 데이터 형식 스트림 읽기 지원 메커니
  - 객체상태를 메모리나 영구 저장 가능한 0과 1의
  - 클래스 선언부에 [Serializable] 애트리뷰트 작성
  - Stream 클래스와 BinaryFormatter 이용
  - 역 직렬화 예
  - 직렬화 하고 싶지 않은 필드: [NonSerialized] 애트리뷰트로 수식




```
[Serializable]
class MyClass
{
    public int myField1;
    [NonSerialized]
    public int myField3;

    public int myField4;
}
```

myField3을 제외한 나머지 필드들만 직렬화됩니다.

- **데모 예제 - Serialization**

Thank You !

이것이 C#이다