Module 5

Lire et Ecrire dans des Fichiers

Sommaire

- Accéder au Système de Fichiers
- Lire et Ecrire dans des Fichiers en utilisant des Flux

Leçon 1: Accéder au Système de Fichiers

- Manipuler des Fichiers
- Lire depuis et Ecrire vers des Fichiers
- Manipuler des Répertoires
- Manipuler des Chemins

Manipuler des Fichiers

Interagir avec les fichiers est une exigence commune pour de nombreuses applications

L'espace de noms **System.IO** contient de nombreuses classes qui simplifient l'interaction avec le système de fichiers, telles que les classes **File** et **FileInfo**

La classe **File** :

- **Copy()**
- **Create()**
- **Delete()**
- **Exists()**

Classe **FileInfo** :

- **CopyTo()**
- **Delete()**
- **Length**
- **Open()**

Lire depuis et Ecrire vers des Fichiers

Lire des données depuis des fichiers

```
string filePath = "myFile.txt";
...
byte[] data = File.ReadAllBytes(filePath); // Binary data.
string[] lines = File.ReadAllLines(filePath); // Lines from file.
string data = File.ReadAllText(filePath); // Entire file.
```

Ecrire des données dans des fichiers

```
string filePath = "myFile.txt";
...
string[] fileLines = {"Line 1", "Line 2", "Line 3"};
File.AppendAllLines(filePath, fileLines); // Append lines.
File.WriteAllLines(filePath, fileLines); // Write lines to new file.
...
string fileContents = "I am writing this text to a file ...";
File.AppendAllText(filePath, fileContents); // Append all text..
File.WriteAllText(filePath, fileContents); // Write all lines to new
file.
```

Manipuler des Répertoires

L'espace de noms System.IO contient les classes Directory et DirectoryInfo afin de simplifier les interactions avec les répertoires

La classe **Directory**

```
string dirPath = @"C:\Users\Student\MyDirectory";
...
Directory.CreateDirectory(dirPath);
Directory.Delete(dirPath);
string[] dirs = Directory.GetDirectories(dirPath);
string[] files = Directory.GetFiles(dirPath);
```

La classe **DirectoryInfo**

```
string dirPath = @"C:\Users\Student\MyDirectory";
DirectoryInfo dir = new DirectoryInfo(dirPath);
...
bool exists = dir.Exists;
DirectoryInfo[] dirs = dir.GetDirectories();
FileInfo[] files = dir.GetFiles();
string fullName = dir.FullName;
```

Manipuler des Chemins

L'espace de noms System.IO contient la classe Path, qui peut contribuer à créer et contrôler les noms de chemin d'accès, indépendants du système de fichiers sous-jacent

La classe **Path** class comprend:

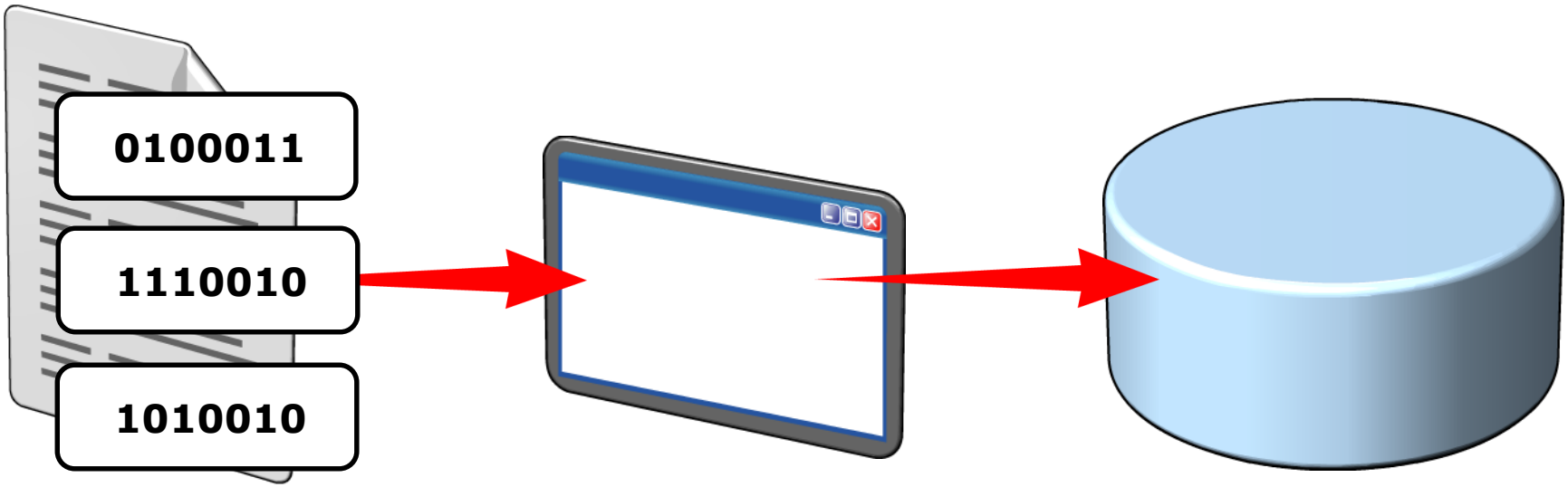
- **GetDirectoryName()**
- **GetExtension()**
- **GetFileName()**
- **GetFileNameWithoutExtension()**
- **GetRandomFileName()**

Leçon 2: Lire et Ecrire dans des Fichiers en utilisant des Flux

- Qu’est-ce que des flux?
- Lire et Ecrire des Données Binaires
- Reading and Writing Text
- Lire et Ecrire des types de données Primitifs

Qu’est-ce que des flux?

Un flux de données est un mécanisme qui permet de manipuler les données en segments gérables



Lire et Ecrire des Données Binaires

```
FileStream sourceFile = new FileStream(sourceFilePath);
BinaryReader reader = new BinaryReader(sourceFile);
int position = 0;
int length = (int)reader.BaseStream.Length;
byte[] dataCollection = new byte[length];
int returnedByte;
while ((returnedByte = reader.Read()) != -1)
{
    dataCollection[position] = (byte)returnedByte;
    position += sizeof(byte);
}
reader.Close();
sourceFile.Close();
```

Classe **BinaryReader**

```
byte[] dataCollection = { 1, 4, 6, 7, 12, 33, 26, 98, 82, 101 };
FileStream destFile = new FileStream(destinationFilePath);
BinaryWriter writer = new BinaryWriter(destFile);
foreach (byte data in dataCollection)
{
    writer.Write(data);
}
writer.Close();
destFile.Close();
```

Classe **BinaryWriter**

Lire et Ecrire du Texte

```
FileStream sourceFile = new FileStream(sourceFilePath);
StreamReader reader = new StreamReader(sourceFile);
StringBuilder fileContents = new StringBuilder();
while (reader.Peek() != -1)
{
    fileContents.Append((char)reader.Read());
}
string data = fileContents.ToString();
reader.Close();
sourceFile.Close();
```

Classe **StreamReader**

```
FileStream destFile = new FileStream("...");
StreamWriter writer = new StreamWriter(destFile);

writer.WriteLine("Hello, this will be written to the file");

writer.Close();
destFile.Close();
```

Classe **StreamWriter**

Lire et Ecrire des types de données Primitifs

```
bool boolValue = reader.ReadBoolean();
byte byteValue = reader.ReadByte();
byte[] byteArrayValue = reader.ReadBytes(4);
char charValue = reader.ReadChar();
...
```

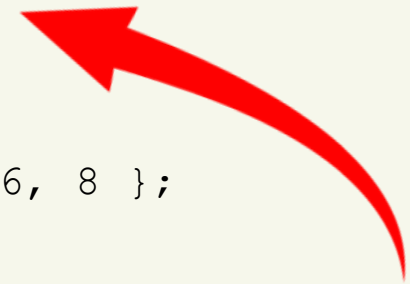
Classe **BinaryReader**

```
bool boolValue = true;
writer.Write(boolValue);

byte byteValue = 1;
writer.Write(byteValue);

byte[] byteArrayValue = { 1, 4, 6, 8 };
writer.Write(byteArrayValue);

char charValue = 'a';
writer.Write(charValue);
...
```



Classe **BinaryWriter**

Lab: Reading and Writing Files

- Exercise 1: Building a Simple File Editor
- Exercise 2: Making the Editor XML Aware

Logon information

Virtual machine	10266A-GEN-DEV
User name	Student
Password	Pa$$w0rd

Estimated time: 45 minutes

Lab Scenario

Lab Review

Review Questions

- Explain the purpose of the **File.Load** and **File.Save** static methods.
- You have a file that contains text. You want to read the file one character at a time. Which method of the **StreamReader** class would you use?

Module Review and Takeaways

- Review Questions
- Best Practices