



L'accès aux fichiers



- Une manière de lire et d'écrire des octets depuis un périphérique de stockage.
 - Les classes Stream héritent de **System.IO.Stream**
- Les méthodes fondamentales des Streams sont **Read, Write, and Seek**:
 - les propriétés **CanRead, CanWrite, et CanSeek**
- Tous les Streams supportent le buffering afin d'améliorer les performances
 - La méthode **Flush** écrit et vide les buffers internes.



- La méthode **Close** libère les ressources:
 - la méthode **Close** appelle implicitement **Flush** pour les streams bufferisés.
- Classes Stream mises à disposition par .NET Framework:
 - **NetworkStream, BufferedStream, MemoryStream, FileStream, CryptoStream**
- Une instance à **null** d'un stream n'a pas de stockage.



- Les classes dérivées de **System.IO.Stream** permettent uniquement de lire et d'écrire des octets.
- Les Readers et Writers permettent de lire et d'écrire d'autres types dans des Streams ou des chaînes de caractères.
- **BinaryReader** et **BinaryWriter** permettent de lire et d'écrire des types primitifs dans un Stream.



- **TextReader** et **TextWriter** sont des types abstraits qui implémentent les méthodes de lecture et d'écriture des caractères.
- Les classes dérivées de **TextReader** et **TextWriter** :
 - **StreamReader** et **StreamWriter**, lecture et écriture dans un stream.
 - **StringReader** et **StringWriter**, lecture et écriture dans une chaîne de caractères ou un **StringBuilder** respectivement.



- La classe **FileStream** est utile pour lire et écrire dans un fichier.
- Les paramètres du constructeur de la classe **FileStream** :
 - **FileMode** – Open, Append, Create
 - **FileAccess** – Read, ReadWrite, Write
 - **FileShare** – None, Read, ReadWrite, Write

```
FileStream f = new FileStream(name, FileMode.Open,  
    FileAccess.Read, FileShare.Read);
```

- Accès direct en utilisant la méthode :
 - Spécifiez l'offset en octet
 - Spécifiez le point relatif de référence pour le saut



- **File** est une classe utilitaire avec des méthodes statiques pour :
 - Créer, copier, effacer, déplacer, et ouvrir des fichiers.
- **FileInfo** est une classe utilitaire avec des méthodes instanciées pour :
 - Créer, copier, effacer déplacer et ouvrir des fichiers.
 - Éliminer quelques contrôles de sécurité en réutilisant l'objet.
- Exemple:
 - Créer **unStream** sur un nouveau fichier nommé foo.txt dans le dossier courant.

```
FileStream unStream = File.Create("foo.txt");
```



Lecture d'un fichier texte

- Lecture d'un fichier texte et affichage sur la console :

```
//...
StreamReader sr = File.OpenText(FILE_NAME);
String input;
while ((input=sr.ReadLine())!=null) {
    Console.WriteLine(input);
}
Console.WriteLine ("Fin de fichier trouvée.");
sr.Close();
//...
```



Écriture d'un fichier texte

- Création du fichier.
- Écriture d'une chaîne, d'un **integer** et d'un **float**.
- Fermeture du fichier.

```
//...  
StreamWriter sw = File.CreateText("MyFile.txt");  
sw.WriteLine ("C'est mon fichier !");  
sw.WriteLine (  
    "J'ai écrit : ints {0} ou floats {1}", 1, 4.2);  
sw.Close();  
//...
```



Directory et DirectoryInfo

- **Directory** dispose de méthodes statiques pour :
 - Créer, déplacer, et énumérer des dossiers et sous-dossiers.
- **DirectoryInfo** dispose de méthodes instanciées pour :
 - Créer, déplacer , et énumérer des dossiers et sous-dossiers.
 - Éliminer quelques contrôles de sécurité en réutilisant l'objet.
- Exemple:

```
DirectoryInfo dir = new DirectoryInfo(".");  
foreach (FileInfo f in dir.GetFiles("*.cs")) {  
    Console.WriteLine(f.FullName); }  
}
```

- Utiliser la classe **Path** pour manipuler des chemins d'accès.



FileSystemWatcher

- **FileSystemWatcher** permet de surveiller un système de fichier
- Création d'un objet **FileSystemWatcher** :

```
FileSystemWatcher watcher = new FileSystemWatcher();
```

- Paramétrage :

```
watcher.Path = ".";  
watcher.Filter = "*.txt";  
watcher.NotifyFilter = NotifyFilters.FileName;  
watcher.Renamed += new RenamedEventHandler(OnRenamed);
```

- Début de la surveillance :

```
watcher.EnableRaisingEvents = true;
```

- Réception des notifications :

```
public static void OnRenamed(object s, RenamedEventArgs e) {  
    Console.WriteLine("Fichier: {0} renommé en {1}",  
        e.OldFullPath, e.FullPath);  
}
```