Лекция № 7

I T E A

IT Education Academy

WWW.ITEA.UA

C# Base

Урок № 7

Программирование Ввода - Вывода

План урока

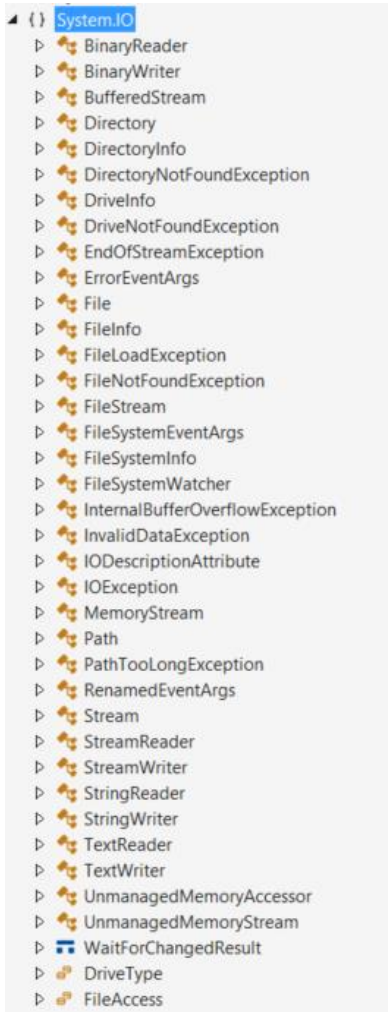
- Навигация по файловой системе
- Чтение и запись файлов
- Работа с потоками
- Библиотека System.IO
- Изолированное хранилище

Работа с потоками и файловой системой

- Большинство задач в программировании связаны с работой с файлами и каталогами. Нам может потребоваться прочитать текст из файла или наоборот произвести запись, удалить файл или целый каталог, не говоря уже о более комплексных задачах, как например, создание текстового редактора и других подобных задачах.

- Фреймворк .NET предоставляет большие возможности по управлению и манипуляции файлами и каталогами, которые по большей части сосредоточены в пространстве имен System.IO. Классы, расположенные в этом пространстве имен (такие как Stream, StreamWriter, FileStream и др.), позволяют управлять файловым вводом-выводом.

Пространство имен - System.IO



Пространство имен **System.IO** содержит типы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также типы для базовой поддержки файлов и папок.

Подобно любому пространству имен, в **System.IO** определен набор классов, интерфейсов, перечислений, структур и делегатов. Многие типы из пространства имен **System.IO** сосредоточены на программных манипуляциях физическими каталогами и файлами. Дополнительные типы предоставляют поддержку чтения и записи данных в строковые буферы, а также области памяти.

Программирование Ввода - Вывода

На диске не может быть двух файлов с одинаковыми именами.

D:\Folder1\Folder2\File.txt

D:\Folder1\Folder2\Folder3\File.txt

Directory and DirectoryInfo

Для работы с каталогами в пространстве имен **System.IO** предназначены два класса: **Directory и DirectoryInfo**. Они содержа в себе группу членов для создания, удаления, перемещения и переименования каталогов и подкаталогов.

Класс **Directory** предоставляет ряд статических методов для управления каталогами. Некоторые из этих методов:

- **CreateDirectory(path)**: создает каталог по указанному пути path
- **Delete(path)**: удаляет каталог по указанному пути path
- **Exists(path)**: определяет, существует ли каталог по указанному пути path. Если существует, возвращается true, если не существует, то false
- **GetFiles(path)**: получает список файлов в каталоге path

Directory and DirectoryInfo

Класс **DirectoryInfo** предоставляет функциональность для создания, удаления, перемещения и других операций с каталогами. Во многом он похож на Directory. Некоторые из его свойств и методов:

- **Create()**: создает каталог
- **CreateSubdirectory(path)**: создает подкаталог по указанному пути path
- **Delete()**: удаляет каталог
- **Свойство Exists**: определяет, существует ли каталог
- **GetDirectories()**: получает список каталогов
- **GetFiles()**: получает список файлов
- **MoveTo(destDirName)**: перемещает каталог
- **Свойство Parent**: получение родительского каталога
- **Свойство Root**: получение корневого каталога

File and FileInfo

Подобно паре Directory/DirectoryInfo для работы с файлами предназначена пара классов **File** и **FileInfo**. С их помощью мы можем создавать, удалять, перемещать файлы, получать их свойства и многое другое.

Методы и свойства класса **FileInfo**:

- **CopyTo(path)**: копирует файл в новое место по указанному пути path
- **Create()**: создает файл
- **Delete()**: удаляет файл
- **MoveTo(destFileName)**: перемещает файл в новое место
- Свойство **Name**: получает имя файла
- Свойство **FullName**: получает полное имя файла

File and FileInfo

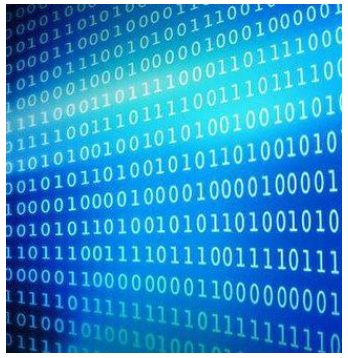
Класс File реализует похожую функциональность с помощью статических методов:

- **Copy()**: копирует файл в новое место
- **Create()**: создает файл
- **Delete()**: удаляет файл
- **Move**: перемещает файл в новое место
- **Exists(file)**: определяет, существует ли файл

FileStream

- Класс **FileStream** представляет возможности по считыванию из файла и записи в файл. Он позволяет работать как с текстовыми файлами, так и с бинарными.

- Это элементарный поток, и он может записывать или читать только один байт или массив байтов. Однако взаимодействовать с членами типа **FileStream** придется нечасто. Вместо этого, скорее всего, будут использоваться оболочки потоков, которые облегчают работу с текстовыми данными или типами .NET.



Поток Байтов

Поток людей

Поток воды

MemoryStream

- **MemoryStream** представляет собой реализацию класса "Stream", в которой массив байтов используется для ввода и вывода.

- Класс "**MemoryStream**" является полезным, когда есть необходимость читать вводимые данные из массива или записывать выводимые данные в массив, а не вводить их непосредственно из устройства или выводить прямо на него.

- Где можно применить запоминающие потока. Например, можно организовать сложный вывод с предварительным накоплением данных в массиве до тех пор, пока они не понадобятся. Также стандартный поток может быть переадресован из массива. Это может понадобиться в том случае, когда необходимо подавать тестовую информацию в программу.

StreamReader и StreamWriter

Для записи в текстовый файл используется класс **StreamWriter**. Некоторые из его конструкторов, которые могут применяться для создания объекта **StreamWriter**.

- **int Close()**: закрывает записываемый файл и освобождает все ресурсы;
- **void WriteLine(string value):** записывает данные, только после записи добавляет в файл символ окончания строки.

Класс **StreamReader** позволяет нам легко считывать весь текст или отдельные строки из текстового файла. Среди методов **StreamReader** можно выделить следующие:

- **void Close()**: закрывает считываемый файл и освобождает все ресурсы;
- **int Peek()**: возвращает следующий символ, если символов больше нет, то возвращает -1;
- **int Read()**: считывает и возвращает следующий символ в численном представлении.
- **string ReadToEnd()**: считывает весь текст из файла;

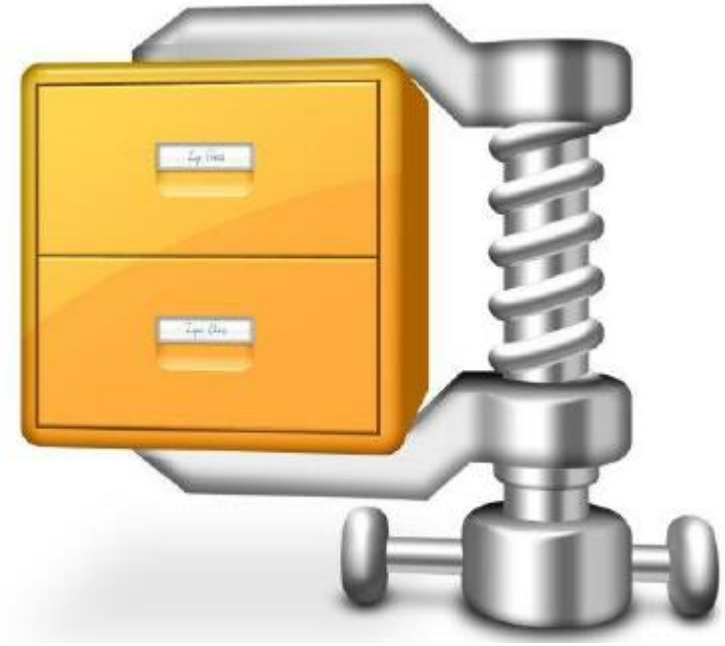
Класс Path

Класс "**Path**" ни с какими папками и файлами не работает. Он работает со строками, в которых записаны пути. Все методы этого класса являются статическими.

Данный класс удобно использовать в некоторых случаях:

- Имеется полный путь до файла: "путь + название файла". Из этого полного пути можно получить короткое имя файла, т.е. только название, использовав метод **GetFileName();**
- Можно получить расширение с помощью метода **GetExtension()**;
- Также можно получить только папку, которая содержит этот файл - метод **GetPathRoot();**
- Особенно данный класс удобно использовать для объединения папки, т.е. пути к ней (или группы папок) и имени файла. Делать это можно с помощью метода **Combine** класса **Path**.

Компрессия / Декомпрессия



Архивация файлов - перекодирование данных с целью уменьшения их объёма.

Разархивация - процесс, обратный архивации, т. е. процесс восстановления данных из архивных файлов.

ZIP — популярный формат сжатия данных и архивации файлов. Файл в этом формате обычно имеет расширение .zip и хранит в сжатом или несжатом виде один или несколько файлов. Термин «ZIP» предлагается интерпретировать как «скорость».

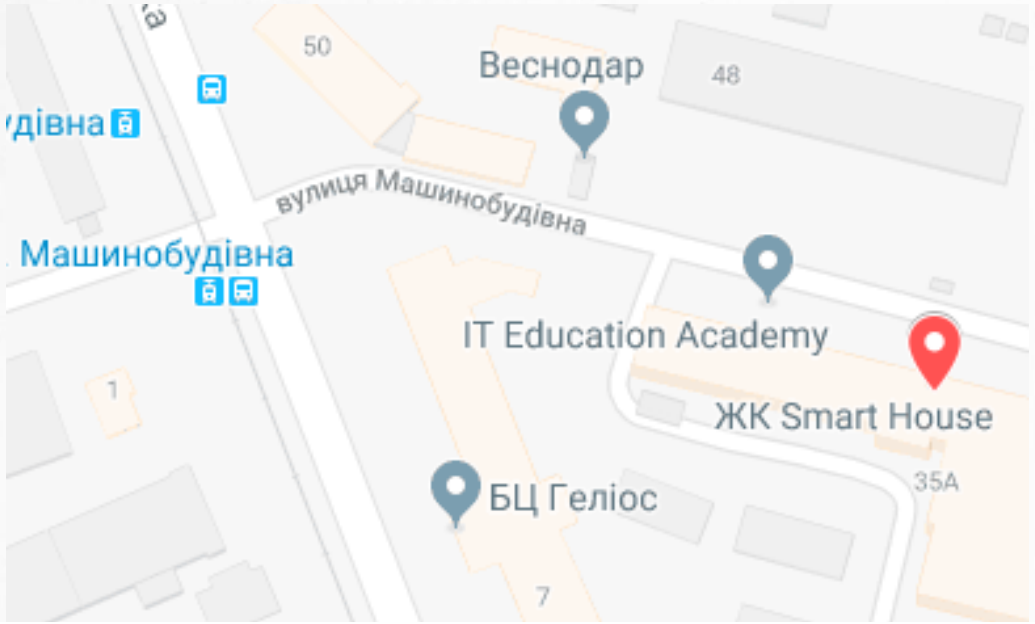
Deflate — это алгоритм сжатия без потерь, который использует комбинацию алгоритма LZ77 и алгоритма Хаффмана.

Isolate Storage

Isolate Storage – Изолированное Хранилище – «Песочница

Песочница (Изолированное Хранилище) – это механизм для безопасного исполнения программ. Как правило, песочницы используют для запуска непроверенного кода из неизвестных источников или для тестирования кода программ, который может случайно повредить всю систему или испортить сложную конфигурацию


Презентация окончена.
Спасибо за внимание!



I T E A

КОНТАКТНЫЕ ДАННЫЕ

ITEA

ЖК “Smart House”, ул. Машиностроительная, 41 (м.Берестейская)

ЖК «Корона» улица Срибнокильская,1 м. Позняки

+38 (044) 599-01-79
facebook.com/Itea
info@itea.ua
itea.ua