FILEINFO Y DIRECTORYINFO

Manipulación de archivos mediante File y FileInfo

Las clases **File** y **FileInfo**, proporcionan a través de sus miembros, el conjunto de operaciones comunes que podemos realizar con archivos en cuanto a su creación, copia, borrado, etc.
La diferencia principal entre ambas radica en que los miembros de File son todos compartidos, con lo cual se facilita en gran medida su uso, al no tener que crear una instancia previa de la clase; mientras que en FileInfo deberemos crear un objeto para poder utilizarla, ya que sus miembros son de instancia.
FileInfo dispone de algunos métodos adicionales que no se encuentran en File.
Comenzando por la clase File, los métodos CreateText( ) y OpenText( ), devuelven respectivamente un objeto StreamWriter y StreamReader, que utilizaremos para escribir y leer en el archivo pasado como parámetro a estos métodos. Con el método Exists( ), comprobamos si existe un determinado archivo. Veamos un ejemplo en el Código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace FileInfo1
{
    class Program
    {
        static void Main(string[] args)
        {
            string sNombreFich;
            StreamReader srLector;
            StreamWriter swEscritor;
            Console.WriteLine("Introducir ruta y archivo");
            sNombreFich = Console.ReadLine();
            if (File.Exists(sNombreFich))
            {
                srLector = File.OpenText(sNombreFich);
                Console.WriteLine("El archivo contiene:{0}{1}", srLector.ReadToEnd());
                srLector.Close();
            }
            else
            {
                swEscritor = File.CreateText(sNombreFich);
                swEscritor.WriteLine("este es");
                swEscritor.WriteLine("un nuevo archivo");
                swEscritor.Close();
            }
            Console.WriteLine("Proceso finalizado");
            Console.ReadLine();

        }
    }
}
```

Para obtener los atributos de un archivo, disponemos del método **GetAttributes( )**, al que pasamos la ruta de un archivo, y devuelve un valor de la enumeración **FileAttributes** con

la información sobre los atributos. En el caso de que al intentar acceder a un archivo, este no exista, se producirá una excepción de tipo **FileNotFoundException**, que podemos tratar en una estructura de manejo de excepciones. Ver el siguiente código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace FileInfo2
{
    class Program
    {
        static void Main(string[] args)
        {
            string sNombreFich;
            FileAttributes oAtributos;
            try
            {
                Console.WriteLine("Introducir ruta y archivo");
                sNombreFich = Console.ReadLine();
                oAtributos = File.GetAttributes(sNombreFich);
                Console.WriteLine("Atributos del archivo: {0}", oAtributos.ToString());
            }
            catch (FileNotFoundException oExcep)
            {
                Console.WriteLine("Se ha producido un error {0}", oExcep.Message);
            }
            finally
            {
                Console.WriteLine("Proceso finalizado");
                Console.ReadLine();
            }
        }
    }
}
```

Los métodos **Copy( ), Move( ) y Delete( ),** nos permiten copiar, mover y borrar respectivamente el nombre de archivo que pasemos como parámetro. El método **GetCreationTime( )** nos devuelve un tipo Date con la fecha de creación del archivo.
Por otro lado, si queremos obtener información adicional sobre un archivo, como su nombre, extensión, ruta, etc., instanciaremos un objeto **FileInfo( )**, pasando al constructor una cadena con el nombre del archivo, y utilizaremos algunas de sus propiedades como **Name, Extensión, DirectoryName**. Veamos una muestra de todo esto:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace FileInfo3
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
             string sNombreFich;
             int iOperacion;
             FileInfo  oFInfo;

            Console.WriteLine("Introducir ruta y archivo");
            sNombreFich = Console.ReadLine();
            Console.WriteLine("Fecha creación archivo:
                        {0}",File.GetCreationTime(sNombreFich));
            oFInfo = new FileInfo(sNombreFich);

            Console.WriteLine("Introducir el número de operación a realizar:");
            Console.WriteLine("1 - Copiar");
            Console.WriteLine("2 - Mover");
            Console.WriteLine("3 - Borrar");
            iOperacion = Convert.ToInt32(Console.ReadLine());
            switch( iOperacion)
            {
                case 1:
                    File.Copy(sNombreFich, ".\\distinto"+ oFInfo.Extension);
                    break;
                case 2:
                    Console.WriteLine("Vamos a mover el archivo {0}", oFInfo.Name);
                    Console.WriteLine("que está en la ruta {0}", oFInfo.DirectoryName);
                    File.Move(sNombreFich, ".\\pruebas\\" + oFInfo.Name);
                    Console.WriteLine("Completado");
                    Console.ReadLine();
                    break;
                case 3:
                    File.Delete(sNombreFich);
                    break;

            }

        }
    }
}
```

Otros ejemplos de utilización del FileInfo:

```
// Este ejemplo muestra el nombre y el tamaño
//de los ficheros de un directorio especificado
using System;
using System.IO;
public class FileLength
{
     public static void Main()
     {
          // Creamos DirectoryInfo para hacer referencia al directorio
           DirectoryInfo di = new DirectoryInfo("c:\\");
          // Hacemos referencia a cada uno de los ficheros de ese directorio
       FileInfo[] fiArr = di.GetFiles();
          // Muestra nombre y tamaño de los ficheros.
          Console.WriteLine("El directorio {0} contiene los  ficheros:", di.Name);
          foreach (FileInfo f in fiArr)
               Console.WriteLine("El tamaño del fichero {0} es {1} bytes.", f.Name,
f.Length);
     }
}
```

```
using System;
using System.IO;
using System.Security;

namespace FileInfo4
{
    class Program
    {
        static void Main(string[] args)
        {
            string path = null;

            Console.WriteLine("Introduce un archivo: ");
            path = Console.ReadLine();
            Console.WriteLine();

            try
            {
                FileInfo file = new FileInfo(path);
                if (file.Exists)
                {
                    Console.WriteLine("Nombre: {0}", file.Name);
                    Console.WriteLine("Nombre completo: {0}", file.FullName);
                    Console.WriteLine("Directorio padre: {0}", file.DirectoryName);
                    Console.WriteLine("Es de lectura: {0}", file.IsReadOnly);
                    Console.WriteLine("Tamaño: {0} bytes", file.Length);
                    Console.WriteLine("Atributos: {0}", file.Attributes);
                    Console.WriteLine("Creado: {0}", file.CreationTime);
                    Console.WriteLine("Leído: {0}", file.LastAccessTime);
                    Console.WriteLine("Modificado: {0}", file.LastWriteTime);
                }
                else
                {
                    Console.WriteLine("El archivo '{0}' no existe. ", path);
                }
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine("El archivo '{0}' es inválido. {1}",path, ex.Message);
            }
            catch (PathTooLongException ex)
            {
                Console.WriteLine("El archivo '{0}' tiene muchos caracteres. {1}",path,
                    ex.Message);
            }
            catch (SecurityException ex)
            {
                Console.WriteLine("No tiene permiso para acceder a '{0}'. {1}",path,
                    ex.Message);
            }
            catch (NotSupportedException ex)
            {
                Console.WriteLine("El archivo '{0}' contiene caracteres inválidos.
                    {1}",path, ex.Message);
            }
            catch (IOException ex)
            {
                Console.WriteLine("El directorio '{0}' no está disponible. {1}",path,
                    ex.Message);
            }

            Console.ReadKey(true);
        }
```

```
    }
}
```

Manipulación de archivos mediante Directory y DirectoryInfo

Las clases **Directory** y **DirectoryInfo** contienen métodos y propiedades para crear, borrar, copiar y mover directorios, así como otra serie de tareas para su manejo y obtención de información.
Al igual que sucedía con las clases del anterior apartado, los miembros de Directory son compartidos, mientras que los de DirectoryInfo son de instancia; esta es su principal diferencia. En el ejemplo del código fuente mostrado a continuación, el método **Exists( )** comprueba la existencia de un directorio, y en caso afirmativo, obtenemos su última fecha de uso mediante **GetLastAccessTime( )**. Seguidamente obtenemos un array String con su lista de archivos mediante **GetFiles( )**, y creamos un subdirectorio de respaldo con **CreateSubdirectory( )**. En caso de que el directorio no exista, lo creamos con **CreateDirectory( )**.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace DirectoryInfo1
{
    class Program
    {
        static void Main(string[] args)
        {
            string sNombreDir;
            string[] Archivos;
            DirectoryInfo  oDInfo;


            Console.WriteLine("Introducir un nombre de directorio");
            sNombreDir = Console.ReadLine();


            if (Directory.Exists(sNombreDir))
            {
                Console.WriteLine("Fecha último acceso:
                    {0}",Directory.GetLastAccessTime(sNombreDir));
                Console.WriteLine("Archivos del directorio {0}", sNombreDir);
                Archivos = Directory.GetFiles(sNombreDir);
                foreach (string Archivo in Archivos){
                    Console.WriteLine(Archivo);
                }

                oDInfo = new DirectoryInfo(sNombreDir);
                oDInfo.CreateSubdirectory("bak");
            }
            else
            {
                Directory.CreateDirectory(sNombreDir);
                Console.WriteLine("No existía el directorio, se acaba de crear");
            }
            Console.ReadKey();
```

```
        }
    }
}
```

Para obtener el directorio actual de ejecución, disponemos del método **GetCurrentDirectory( )**, mientras que si queremos subir al directorio de nivel superior, tenemos el método **GetParent( )**, que devuelve un tipo DirectoryInfo, con el que podemos, por ejemplo, mostrar su nombre completo mediante la propiedad **FullName**, y fecha de creación con **CreationTime**. Veamos el Código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace DirectoryInfo2
{
    class Program
    {
        static void Main(string[] args)
        {
            string sNombreDir;
            DirectoryInfo oDirInfo;

            //obtenemos el directorio actual de ejecución
            sNombreDir = Directory.GetCurrentDirectory();
            Console.WriteLine("Directorio actual: {0}", sNombreDir);

            //obtenemos el directorio padre del actual, y mostramos información de dicha
            //directorio
            oDirInfo = Directory.GetParent(sNombreDir);
            Console.WriteLine("Directorio padre y fecha de creación {0}\n{1}",
                              oDirInfo.FullName, oDirInfo.CreationTime);

            Console.ReadKey();
        }
    }
}
```

En el siguiente ejemplo, el método GetDirectories( ) devuelve un array de cadenas, con los nombres de los subdirectorios que se encuentran dentro del directorio pasado como parámetro a este método. A continuación, mediante el método Move( ), cambiamos de lugar un directorio; con Delete( ) borramos otro de los directorios. Observe el lector, cómo utilizando de forma combinada CType( ), Directory.GetFiles( ), y un elemento del array que contiene la lista de directorios, creamos una expresión que nos permite averiguar, si en un determinado directorio hay o no archivos. Ver el Código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;


namespace DirectoryInfo3
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
             string sNombreDir;
             DirectoryInfo oDirInfo;
             string[] sDirectorios;


            Console.WriteLine("Introducir un nombre de directorio");
            sNombreDir = Console.ReadLine();

            //obtener directorios del directorio especificado
            sDirectorios = Directory.GetDirectories(sNombreDir);

            //comprobar que el directorio contiene a su vez
            //varios directorios; en caso negativo, finalizar
            if (sDirectorios.Length < 1)
            {
                Console.WriteLine("El directorio especificado debe contener al menos dos
subdirectorios");
                Console.ReadLine();


            }
            else
            {
                //mostrar nombres de directorios

                foreach (string sDirectorio in sDirectorios)
                {
                    Console.WriteLine(sDirectorio);
                }
                //mover uno de los directorios a otra ubicación del disco actual
                Directory.Move(sDirectorios[0], ".\\temp\\BIS");

                //borrar otro de los directorios
                //el directorio a borrar debe estar vacío
                //comprobar con la siguiente expresión si dicho
                //directorio contiene o no archivos
                string[] archivos = Directory.GetFiles(sDirectorios[1]);

                if (archivos.Length >1)
                {
                    Console.WriteLine("No se puede borrar el directorio: {0} - " + "contiene
                     archivos", sDirectorios[1]);
                }
                else
                {
                    Directory.Delete(sDirectorios[1]);
                }
                Console.WriteLine("Completado");
                Console.ReadLine();
            }
        }
    }
}
```

La clase Path

Esta clase nos proporciona un conjunto de campos y métodos compartidos, para la

obtención de información y manipulación de rutas de archivos. El código fuente incluido a continuación, muestra un ejemplo en el que, una vez introducido un directorio, se muestra la información de cada uno de sus archivos, en lo que respecta a los métodos de esta clase.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace DirectoryInfo4
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Introducir nombre de directorio");
            string sDirectorio;
            sDirectorio = Console.ReadLine();
            string[] sArchivos;
            sArchivos = Directory.GetFiles(sDirectorio);

            Console.WriteLine("Datos sobre archivos obtenidos del objeto Path");
            Console.WriteLine("============================================");

            foreach (string sArchivo in sArchivos){
                Console.WriteLine("GetDirectoryName() {0}",
                    Path.GetDirectoryName(sArchivo));
                Console.WriteLine("GetExtension() {0}", Path.GetExtension(sArchivo));
                Console.WriteLine("GetFileName() {0}", Path.GetFileName(sArchivo));
                Console.WriteLine("GetFileNameWithoutExtension()
                    {0}",Path.GetFileNameWithoutExtension(sArchivo));
                Console.WriteLine("GetFullPath() {0}", Path.GetFullPath(sArchivo));
                Console.WriteLine();
            }
            Console.ReadLine();
        }
    }
}
```