文件流FileStream类

含FileStream(操作字节)和 StreamReader、StreamWriter(操作字符)两大类

一、FileStream (操作字节)

1.读取数据

```
    //一、使用FileStream读取数据
    //创建FileStream对象
    /**
    * FileStream(string path,FileMode mode,FileAccess access)
    * @path : 要读取文件的路径
    * @FileMode : 针对文件要进行的操作
    * @FileAccess : 针对文件内的数据要进行的操作
    */
    FileStream fsRead = new
FileStream(@"C:\Users\Admin\Desktop\new.txt",FileMode.OpenOrCreate,FileAccess.Read
);

    byte[] buffer = new byte[1024*1024*5];  //用于临时存放读取的数据，数组长度就是单
次读取的最大字节数

    /** 读取以流的方式读取文件数据
    * Read(byte[] buffer,int offset,int count)
    * @buffer : 字节数组，用于缓存从流中读取的字节数据
    * @offset : int型，用于指定从第几个字节开始读取
    * @count : int型，用于指定从流中单次读取的最大字节数
    * 注：若count > 数据流总长度，则不足之处会补0
    * return : 返回本次实际读取到的有效字节数
    */
    int r = fsRead.Read(buffer,0,buffer.Length);

    /** 将字节数组中每一个元素按照指定的编码格式解码成字符串
    * Encoding.Default.GetString(byte[] bytes,int index,int count)
    * @Encoding.Default : 以默认编码形式解码
    * @bytes : 被解码的字节数组
    * @index : 解码起始位置
    * @count : 解码的总长度
    * return : 解码完后返回一个字符串
    */
    string s = Encoding.UTF8.GetString(buffer,0,r);
    //关闭流
    fsRead.Close();
    //释放流所占用的资源
    fsRead.Dispose();
    System.Console.WriteLine(s);
```

注：将创建文件流对象的过程写在using中，会在自动释放文件流所占用的资源
using( ){ } : 小括号中写创建过程，大括号中写操作过程

2.写入数据

```
    //二、使用FileStream写入数据
    using(FileStream fsWrite = new
FileStream(@"C:\Users\Admin\Desktop\newNew.txt",FileMode.OpenOrCreate,FileAccess.W
rite))
    {   //使用using(){}不用手动关闭和释放资源
        //准备写入的数据
        string str = "使用FileStream的Write方法写入数据，看我会不会把原数据覆盖掉";
        //将要写入的数据一UTF8编码的格式转换成字节数组
        byte[] writeBuffer = Encoding.UTF8.GetBytes(str);

        /**
        * Write(byte[] buffer,int offset,int count)
        * 注：该方法会从前往后将原数据覆盖掉
        */
        fsWrite.Write(writeBuffer,0,writeBuffer.Length);
        System.Console.WriteLine("写入完成");
    }
```

3.复制、剪切、粘贴练习

```
    //创建一个文件操作类
    public class OperateFile{

        //用于复制文件的静态方法
        public static void CopyFile(string source,string target){
            //1.创建一个负责读取的流
            using(FileStream fsRead = new
FileStream(source,FileMode.Open,FileAccess.Read)){
                //2.创建一个负责写入的流（必须嵌套在读取流的里面，不然还没写入，using()
就把读取的资源释放掉了）
                using(FileStream fsWrite = new
FileStream(target,FileMode.OpenOrCreate,FileAccess.Write)){
                    byte[] buffer = new byte[1024*1024*5];
                    //因为文件可能比较大，因此应该循环读取
                    while(true){
                        //读取并返回本次实际读取到的字节数
                        int r = fsRead.Read(buffer,0,buffer.Length);
                        //若全部数据都已经读取完毕则跳出死循环
                        if(r == 0){
                            break;
                        }
                        //将读取到的数据写入
                        fsWrite.Write(buffer,0,r);
                    }
                }
```

```
            }
        }

    }

    static void Main(string[] args){
        //复制粘贴
        string source =@"C:\Users\Admin\Desktop\MASTA培训资料.mp4";
        string target =@"C:\Users\Admin\Desktop\myNew.mp4";
        OperateFile.CopyFile(source,target);
        System.Console.WriteLine("复制成功！");

        //剪切粘贴，即复制后删除原文件
        //使用File.Delete(string path)删除原文件
    }
```

二、treamReader和StreamWriter (操作字符)

1.StreamReader读取文本

```
    //使用StreamReader来读取一个文本文件
    using(StreamReader sr = new
StreamReader(@"C:\Users\Admin\Desktop\new.txt",Encoding.UTF8)){

        //sr.EndOfStream表示流是否读取完成，完成返回true，否则返回false。在FileStream
中没有本方法
        while(!sr.EndOfStream){
            //ReadLine()逐行读取
            System.Console.WriteLine(sr.ReadLine());
        }
    }
```

2.StreamWriter写入文本

```
    //使用StreamWriter来写入一个文本文件
    //StreamWriter(string path)     覆盖写入
    //StreamWriter(string path,bool append,Encoding encoding)       不覆盖原数据而
是追加写入
    using(StreamWriter sw = new
StreamWriter(@"C:\Users\Admin\Desktop\new.txt",true,Encoding.UTF8)){
        sw.Write("今天天气很晴朗");
    }
```