Чтение и запись текстовых файлов. StreamReader и StreamWriter

Класс FileStream не очень удобно применять для работы с текстовыми файлами. К тому же для этого в пространстве System.IO определены специальные классы: **StreamReader** и **StreamWriter**.

Запись в файл и StreamWriter

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

- StreamWriter(string path): через параметр path передается путь к файлу, который будет связан с потоком
- StreamWriter(string path, bool append): параметр append указывает, надо ли добавлять в конец файла данные или же перезаписывать файл. Если равно true, то новые данные добавляются в конец файла. Если равно false, то файл перезаписываетсяя заново
- StreamWriter(string path, bool append, System.Text.Encoding encoding): параметр encoding указывает на кодировку, которая будет применяться при записи

Свою функциональность StreamWriter реализует через следующие методы:

- int Close(): закрывает записываемый файл и освобождает все ресурсы
- void Flush(): записывает в файл оставшиеся в буфере данные и очищает буфер.
- Task FlushAsync(): асинхронная версия метода Flush
- void Write(string value): записывает в файл данные простейших типов, как int, double, char, string и т.д. Соответственно имеет ряд перегруженных версий для записи данных элементарных типов, например, Write(char value), Write(int value), Write(double value) и т.д.
- Task WriteAsync(string value): асинхронная версия метода Write
- void WriteLine(string value): также записывает данные, только после записи добавляет в файл символ окончания строки
- Task WriteLineAsync(string value): асинхронная версия метода WriteLine

Рассмотрим запись в файл на примере:

```
using System;
using System.IO;

namespace HelloApp
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            string writePath = @"C:\SomeDir\hta.txt";

            string text = "Привет мир!\nПока мир...";
            try
            {
                using (StreamWriter sw = new
StreamWriter(writePath, false, System.Text.Encoding.Default))
                {
                    sw.WriteLine(text);
                }

                using (StreamWriter sw = new
StreamWriter(writePath, true, System.Text.Encoding.Default))
                {
                    sw.WriteLine("Дозапись");
                    sw.Write(4.5);
                }
                Console.WriteLine("Запись выполнена");
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

В данном случае два раза создаем объект StreamWriter. В первом случае если файл существует, то он будет перезаписан. Если не существует, он будет создан. И в нее будет записан текст из переменной text. Во втором случае файл открывается для дозаписи, и будут записаны атомарные данные - строка и число. В обоих случаях будет использоваться кодировка по умолчанию.

По завершении программы в папке C://SomeDir мы сможем найти файл hta.txt, который будет иметь следующие строки:

```
Привет мир!
Пока мир...
Дозапись
4,5
```

Поскольку операции с файлами могут занимать продолжительное время, то в общем случае рекомендуется использовать асинхронную запись. Используем асинхронные версии методов:

```
using System;
using System.IO;
using System.Threading.Tasks;

namespace HelloApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            string writePath = @"C:\SomeDir\hta2.txt";

            string text = "Привет мир!\nПока мир...";
            try
            {
                using (StreamWriter sw = new
StreamWriter(writePath, false, System.Text.Encoding.Default))
                {
                    await sw.WriteLineAsync(text);
                }

                using (StreamWriter sw = new
StreamWriter(writePath, true, System.Text.Encoding.Default))
                {
                    await sw.WriteLineAsync("Дозапись");
                    await sw.WriteAsync("4,5");
                }
                Console.WriteLine("Запись выполнена");
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Асинхронные версии есть не для всех перегрузок метода Write.

Чтение из файла и StreamReader

Класс StreamReader позволяет нам легко считывать весь текст или отдельные строки из текстового файла.

Некоторые из конструкторов класса StreamReader:

- StreamReader(string path): через параметр path передается путь к считываемому файлу
- StreamReader(string path, System.Text.Encoding encoding): параметр encoding задает кодировку для чтения файла

Среди методов StreamReader можно выделить следующие:

- void Close(): закрывает считываемый файл и освобождает все ресурсы
- int Peek(): возвращает следующий доступный символ, если символов больше нет, то возвращает -1
- int Read(): считывает и возвращает следующий символ в численном представлении. Имеет перегруженную версию: Read(char[] array, int index, int count), где array - массив, куда считываются символы, index - индекс в массиве array, начиная с которого записываются считываемые символы, и count - максимальное количество считываемых символов
- Task<int> ReadAsync(): асинхронная версия метода Read
- string ReadLine(): считывает одну строку в файле
- string ReadLineAsync(): асинхронная версия метода ReadLine
- string ReadToEnd(): считывает весь текст из файла
- string ReadToEndAsync(): асинхронная версия метода ReadToEnd

Сначала считаем текст полностью из ранее записанного файла:

```
using System;
using System.IO;
using System.Threading.Tasks;

namespace HelloApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            string path = @"C:\SomeDir\hta.txt";

            try
            {
                using (StreamReader sr = new StreamReader(path))
                {
                    Console.WriteLine(sr.ReadToEnd());
                }
                // асинхронное чтение
                using (StreamReader sr = new StreamReader(path))
                {
                    Console.WriteLine(await sr.ReadToEndAsync());
                }
            }
            catch (Exception e)
            {
```

```
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Считаем текст из файла построчно:

```
string path= @"C:\SomeDir\hta.txt";

using (StreamReader sr = new StreamReader(path,
System.Text.Encoding.Default))
{
    string line;
    while ((line = sr.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
// асинхронное чтение
using (StreamReader sr = new StreamReader(path,
System.Text.Encoding.Default))
{
    string line;
    while ((line = await sr.ReadLineAsync()) != null)
    {
        Console.WriteLine(line);
    }
}
```

В данном случае считываем построчно через цикл while: while ((line = sr.ReadLine()) != null) - сначала присваиваем переменной line результат функции sr.ReadLine(), а затем проверяем, не равна ли она null. Когда объект sr дойдет до конца файла и больше строк не останется, то метод sr.ReadLine() будет возвращать null.