Всем привет!

Эта статья про использование небольшой библиотеки, которая позволяет делать Linq запросы, для чтения таблицы из файлов.

Набор поддерживаемых форматов файлов легко расширяется реализуя простой интерфейс.

Раньше, в каждом проекте, приходилось импортировать табличные данные из разных файлов (как правило .txt, .csv, .xlsx). Задача всегда сводится к нескольким шагам: вытащить строки, отфильтровать, обработать.

Код для этих операций довольно простой и, поэтому всегда без задней мысли, писался заново.

В какой-то момент я созрел для написания библиотеки - TableReaderLib.

Сейчас реализована поддержка для формата csv (с любыми разделителями, кодировками) и Excel (Необходим Excel на компьютере).

Фишки которые предоставляет библиотека:

- IEnumerable по строкам. Это дает возможность использовать Linq, foreach и ToArray()

- Определять столбцы в виде объектов (это сильно повышает читабельность кода)

- Указывать является ли первая строка заголовками и получить их содержимое

- Если таблица начинается где-то в середине файла, вопрос решается строчкой StartRow= N

- Можно пропускать первые M строк

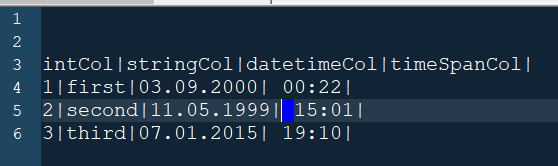
- Можно считывать только K строк

- Реализовать специфические свойства, например, для текстовых файлов это кодировка и тип разделителя, для excel чтение блоками (что весьма ускоряет производительность) и преобразование Excel типов в .net'овские.

Прочитал этой библиотекой, с удовольствием, уже не один миллион строк, и решил показать её миру.

И так давайте попробуем достать табличку из простого текстового файла:

[ATTACH]5345[/ATTACH]



В файле первые две строчки пропущены, что бы показать что делать если таблица начинается не с начала файла.

Затем идёт строка заголовков столбцов. Название заголовка, для наглядности, это тип данных в нём.

Далее три строчки с данными, разделителем является "|", разделитель может быть любым.

Итак, в Visual Studio создаем новый консольный проект, добавляем references:

- TableReaderLib.dll, содержит все общие классы для работы с таблицами, и определение интерфейса ISourceReader, который нужно реализовать для чтения конкретных типов файлов

- ReadersForTableReaderLib.dll - содержит реализации ISourceReader, которые не требуют дополнительных references. (Excel, например, требует)

Затем в коде нам нужно создать объект CsvSourceReader. Конструктор содержит всего три аргумента:

- filePath - путь к нашему файлу

- splitter - разделитель ячеек, в нашем случае "|"

- encoding - кодировка файла, почти всегда можно оставлять null

Объявление конструктора:

[CSHARP]

public CsvSourceReader(string filePath, string splitter = null, Encoding encoding = null)

[/CSHARP]

Затем нужно создать набор столбцов которые нам нужны из исходной таблицы. Для простоты не будем создавать переменную для каждого столбца а сразу инициализируем их в массив:

[CSHARP]

var columns = new TableColumn[]

{

new TableColumn(){IndexInSource = 0 },

new TableColumn(){IndexInSource = 1 },

new TableColumn(){IndexInSource = 2 },

new TableColumn(){IndexInSource = 3 },

};

[/CSHARP]

Вообще в классе TableColumn всего три члена - свойства:

- Name - строковое имя столбца, если для столбца не создавалась переменная, можно доставать ячейки по этому имени

- Type - Тип данных в ячейке. Если быть честным, не придумал как вытаскивать значения с помощью этого свойства. Используются перегруженные методы T GetCellValue<T>(TableColumn)

- IndexInSource - номер столбца в исходной таблице от 0.

Их объявления:

[CSHARP]

public string Name {get; set;}

public Type Type { get; set; }

public int IndexInSource { get; set; }

[/CSHARP]

Теперь создаем объект TableReader, передаем в аргументы конструктора наш CsvSourceReader и Набор столбцов. Затем указываем, с какой строки в файле начинается наша таблица, и что в таблице первая строка является заголовками:

[CSHARP]

var table = new TableReader(reader, columns);

table.StartRow = 2;

table.IsFirstRowHeaders = true;

[/CSHARP]

Настройка готова!

Давайте прочитаем заголовки нашей таблицы в файле через свойство TableReader.SourceHeaders и посмотрим данные из второй строки:

[CSHARP]

string headers = string.Join(";\t",table.SourceHeaders);

var row = table.ToArray()[2];

string rowInfo =

$"column 1 value:{row.GetCellValue<int>(0)}, type:{row.GetCellValue<int>(0).GetType()}" + Environment.NewLine +

$"column 2 value:{row.GetCellValue<string>(1)}, type:{row.GetCellValue<string>(1).GetType()}" + Environment.NewLine +

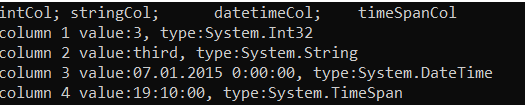
$"column 3 value:{row.GetCellValue<DateTime>(2)}, type:{row.GetCellValue<DateTime>(2).GetType()}" + Environment.NewLine +

$"column 4 value:{row.GetCellValue<TimeSpan>(3)}, type:{row.GetCellValue<TimeSpan>(3).GetType()}" + Environment.NewLine;

[/CSHARP]

Результат:

[ATTACH]5346[/ATTACH]



Для удобства, весь код примера целиком:

[CSHARP]

var reader = new CsvSourceReader(@"D:\myCsv.csv", "|");

var columns = new TableColumn[]

{

new TableColumn(){IndexInSource = 0 },

new TableColumn(){IndexInSource = 1 },

new TableColumn(){IndexInSource = 2 },

new TableColumn(){IndexInSource = 3 },

};

var table = new TableReader(reader, columns);

table.StartRow = 2; //пустые строки перед началом нашей таблицы в файле

table.IsFirstRowHeaders = true; //таблица имеет строку заголовков

string headers = string.Join(";\t",table.SourceHeaders);

var row = table.ToArray()[2];

string rowInfo =

$"column 1 value:{row.GetCellValue<int>(0)}, type:{row.GetCellValue<int>(0).GetType()}" + Environment.NewLine +

$"column 2 value:{row.GetCellValue<string>(1)}, type:{row.GetCellValue<string>(1).GetType()}" + Environment.NewLine +

$"column 3 value:{row.GetCellValue<DateTime>(2)}, type:{row.GetCellValue<DateTime>(2).GetType()}" + Environment.NewLine +

$"column 4 value:{row.GetCellValue<TimeSpan>(3)}, type:{row.GetCellValue<TimeSpan>(3).GetType()}" + Environment.NewLine;

Console.WriteLine(headers);

Console.WriteLine(rowInfo);

Console.ReadLine();

[/CSHARP]

Весь исходный код можно скачать на GitHub и использовать как хотите: [URL="https://github.com/setood/TableReaderLib"]https://github.com/setood/TableReaderLib[/URL]

Буду рад, если кому-то библиотека поможет, обязательно сообщите. Это сильно мотивирует:)

Так же рад замечаниям, code review и помощь в развитии либы.