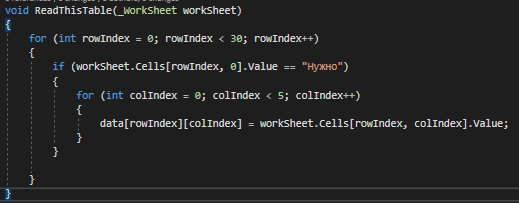
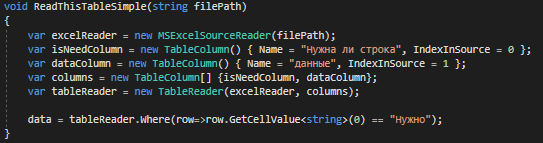
LINQ для таблиц. По строкам на новом велосипеде!

Статья для тех, кому нужно одним движением руки прочитать данные из таблицы.

В своей работе постоянно приходится считывать данные из таблиц из текстовых файлов (\*.csv, \*.txt) и MS Excel (\*.xlsx). Это все делается при помощи большого количества рутинного кода: определить заголовки, сколько строк посмотреть, какие разделители ячеек, цикл по строкам (а в нем цикл по столбцам) и т.д. Кроме того, каждый раз как в том анекдоте «Но есть нюанс» - встречаются какие-то особенности. Поэтому решил написать библиотеку и совершить переход от сложного вроде:



К простому:



Структура Solution

При создании библиотеки хочется уменьшить количество внешних зависимостей, поэтому весь общий функционал нужно поместить в одной библиотеке, а реализации её интерфейсов уже в дополнительных модулях.

И так, имеем:

***TableReaderLib*** – проект в котором находится классы для работы с таблицей (TableReader, TableRow, TableColumn), и интерфейс ISourceReader, который нужно реализовать, чтобы извлечь таблицу из какого-либо источника данных).

***ExcelReaderForTableReaderLib*** – реализация ISourceReader для Excel с использованием Excel.Interop.

***ReadersForTableReaderLib*** – содержит реализацию ISourceReader для текстовых файлов (и остальных видов файлов, не требующих отдельных References, которые появятся когда-нибудь).

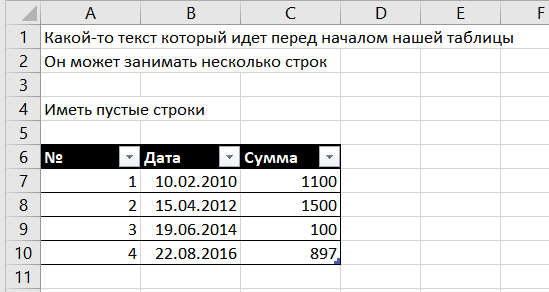
***TableReaderTest*** – тесты.

И так, начнем с основы – проекта TableReaderLib, описания его классов.

Класс TableReader

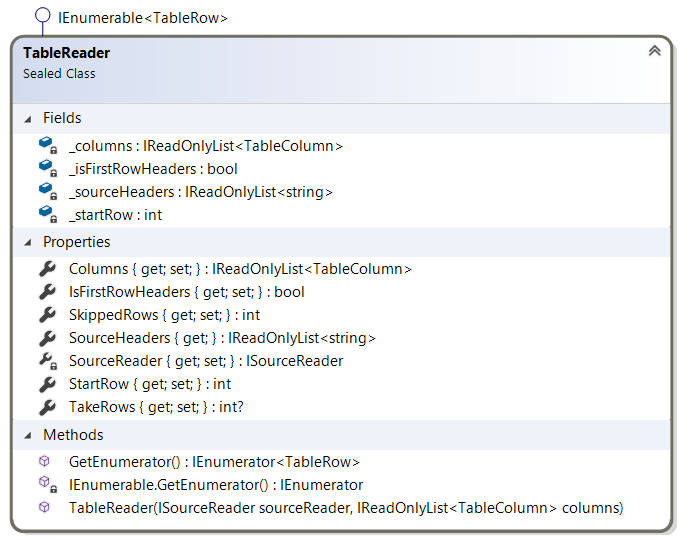
TableReader – ядро библиотеки. Класс должен реализовать интерфейс IEnumerable<TableRow>, что и позволит использовать на нем LINQ запросы. В TableReader хочется иметь функционал, который через изменение свойств, позволит:

* Изменять набор столбцов, из которых нужно получать данные (Columns);
* Отмечать является ли первая строка заголовками таблицы (IsFirstRowHeaders);
* Получать список заголовков SourceHeaders
* Устанавливать строку-начало таблицы (StartRow). Например, на такой случай:



* Возможность пропустить N первых строк (SkippedRows);
* Ограничить количество строк для чтения (TakeRows);
* Конструктор класса должен принимать интерфейс ISourceReader и IReadOnlyList<TableColumn>. ISourceReader – реализует логику чтения из источника, IReadOnlyList<TableColumn> – описывает набор столбцов таблицы для чтения.

На диаграмме классов всё это выглядит так:

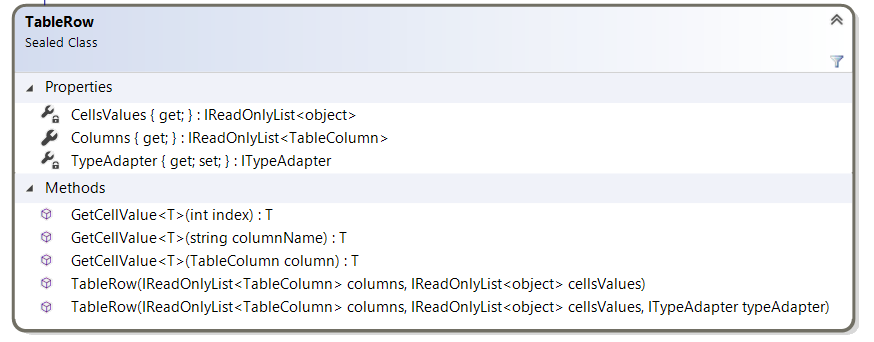


Класс TableRow

Класс является абстракцией строки таблицы и по сути содержит:

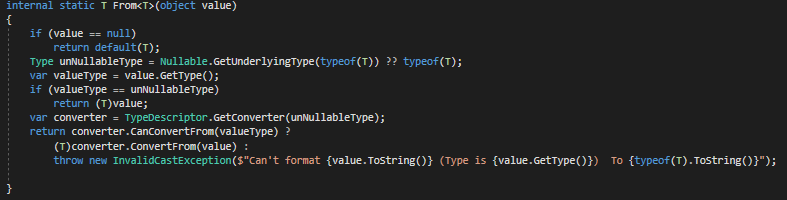
* Набор данных из ячеек строки: CellsValues.
* Методы, позволяющих получить значение ячейке в определенном типе GetCellValue.
* Конструкторы класса.

Диаграмма классов:



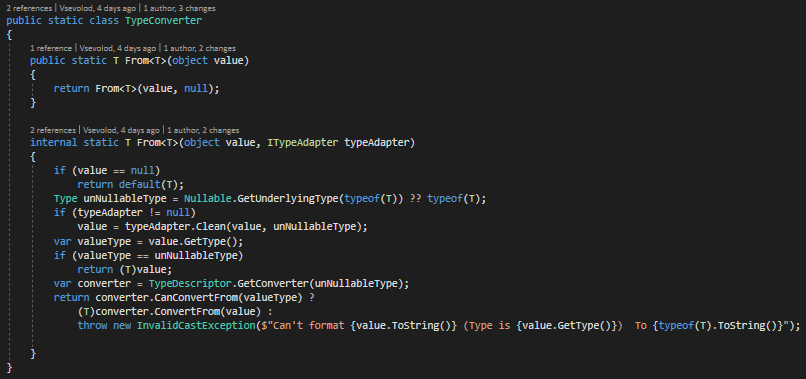
Метод GetCellValue с аргументами (string columnName) и (TableColumn column) сводятся к определению аргумента для метода T GetCellValue<T> (int index), этот метод берет object значение по индексу из CellsValues и используя TypeConverter возвращает его в типе T.

TypeConverter – класс с единственным методом, который позволяет преобразовать object в T:



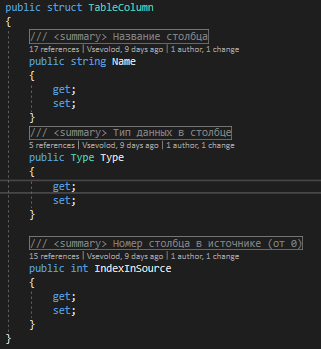
Работая над поддержкой excel, стало понятно, что таким преобразованием типов не обойтись, например, дата там в виде числа, а целых чисел нет вообще. Поэтому появился интерфейс ITypeAdapter, который позволит реализациям ISourceReader, при необходимости, реализовать свое преобразование типов.

В итоге TypeConverter – стал выглядеть так:



Структура TableColumn

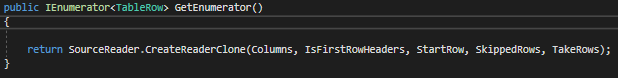
TableColumn – структура для более удобочитаемой работы со столбцами и состоит из нескольких свойств:



Интерфейс ISourceReader

Интерфейс нужен, что бы наш класс TableReader мог работать с источниками данных, которые его реализуют. Интерфейс ISourceReader расширяет IEnumerator<TableRow>, другими словами наши реализации ISourceReader являются итератором по набору строк в источнике, а TableReader реализует поверх него дополнительную Функциональность.

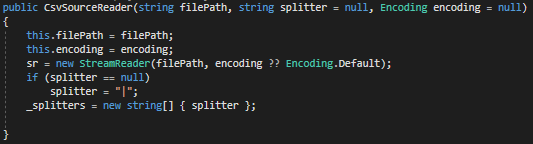
Для одного TableReader может существовать несколько инумераторов, поэтому в интерфейсе есть метод CreateReaderClone, который решает эту проблему. Использование в коде TableReader выглядит так:



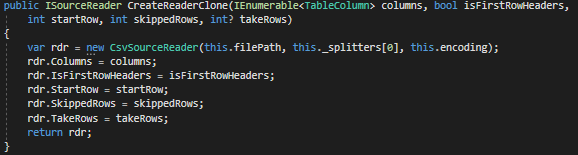
Теперь давайте попробуем реализовать этот интерфейс для csv таблиц.

CsvSourceReader

В основе нашего класса будет лежать обычный StreamReader, поэтому в аргументах конструктора появляются путь к файлу и Encoding. Так же в csv в качестве разделителей ячеек могут быть разные символы, поэтому будем использовать строковый тип. В итоге получаем вот такой конструктор:

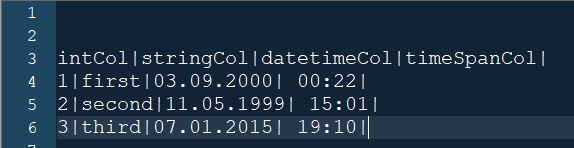


Остается реализовать CreateReaderClone:

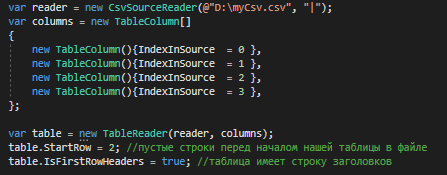


Наконец, можно протестировать что у нас вышло!

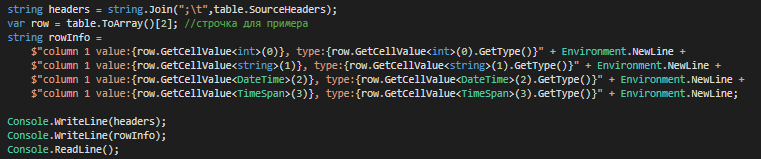
Создаем вот такой файлик:



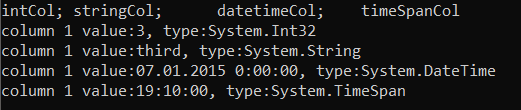
Настраиваемся на чтение этой таблицы:



И пробуем посмотреть заголовки и данные из третьей строки:



И результат:



В проекте на гитхабе уже реализована поддержка Excel, через Interop. Там код довольно сложный и его описание наберется на отдельную статью. Если кому интересно, как ускорить чтение через Excel.Interop пишите, возможно соберусь с мыслями это сделать.

Спасибо за внимание!

Весь проект на гитхабе: <https://github.com/setood/TableReaderLib>