**Upload Parser (XLS/CSV Parser)**

# Description

Upload Parser is a library that can be used to parse an uploaded file. The parser receives the uploaded file stream along with some added parameters from the program, and returns a class object containing complete file data.

Following features are provided by the library:

* Parse entire file and return a class with complete file data encapsulated in it.
* An option for developer to save the uploaded file on the system/server or not.
* An option for developer to set their own path where the uploaded files should be saved.

Currently the plugin supports excel files(.xls) and CSV(.csv) Files. Its functionality can be extended in future to support more file types.

# Plugin Structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Plugin** | **Value** | **Description** | **Usage Example** |
| **Namespace** | UploadParser | To be added with other namespaces | using UploadParser; |
| **Class** | TriggerParse | To be used to create an object which in turn calls the required function | TriggerParse triggerParse = new TriggerParse() |
| **Method** | parseFile | To be called to parse the uploaded file | triggerParse. parseFile(uploadStream, true, @"C:\\MyLoadDir\") |
| **Class** | ExcelFile | An Object class object holds the data of the parsed excel file inits format | Object xlsFile = parseFile (…); |
| **Class** | CSVFile | An Object class object holds the data of the parsed csv file inits format | Object csvFile = parseFile (…); |

### Input Method Description

Object **parseFile** (**Stream** fileUpload, **bool** save, **string** path)

fileUpload: Stream class object. A file uploaded from web, is received at the server end as a Stream

object. Thus, this object is directly passed to the plugin for parsing.

save: A bool value to indicate whether the program wishes to save the uploaded file on the

system/server or not.

path: A string value that allows the program to set a path where the uploaded file will be saved.  
In case of an empty string, the file is saved in the default path set for the same.

### Output Class Description

|  |  |  |
| --- | --- | --- |
| **Class** | **Members** | |
| Object | If Uploaded File type = Excel (.xls) public string fileName;  public string[] sheetName;  public string[][][] sheetData | If Uploaded File type = CSV (.csv) public string fileName;  public string[][] csvData |

fileName: Name of the file uploaded

sheetName: Array of string containing sheet names of all shhets in the excel file

sheetData: A 3 dimensional array containing complete sheet data, i.e.  
 sheetData [sheet index] [row index] [column index]

csvData: A 2 dimensional array containing complete csv data, i.e.  
 csvData [row index] [column index]

# Plugin Usage

Plugin can be used in a code by implementing the following steps:

1. Add a reference of the library (.dll) file in the program
2. Add a reference of the package : NPOI.dll in the program (to support xls workbook operations)
3. Include the **UploadParser** namespace in the code,

i.e.: using UploadParser;

1. Create an object of the class : **TriggerParser**,

i.e.: TriggerParser <myTriggerObjectName> = new TriggerParser();

1. Call the method **parseFile**() to trigger the parsing and get the desired result in Object type ,

i.e.:

Object <myExcelObjectName >= triggerParse.parseFile (<streamObj>, <save?>, <pathToSave>)

## Code Example:

using UploadParser;

...

// Funtion in main program to call the plugin

public string parseUploadedFie(Stream xlsUpload)

{

...

TriggerParse triggerParse = new TriggerParse();

Object File = triggerParse.parseFile(fileUpload,true,"");

...

}

# Plugin Defaults

1. **Default directory to save uploaded file**

UploadParser defines the default path to save the uploaded file to the server, in case the user passes an empty string to the called method.

This default value is :

public const string DEFAULT\_SAVE\_PATH = @"C:\\UPLOAD\_PARSER\";

Thus, if save= true and path = "", the uploaded file will be save to C:\\UPLOAD\_PARSER\ directory.

To set own value, user would be required to pass the value of the desired path in the function :

triggerParse.parseFile (Stream fileUpload, bool save, string path)

For Ex : Object obj = triggerParse.parseFile(csvUpload, true, @"C:\\Loadload\");

1. **Default number of empty rows until the parser stops parsing the file and accepts EOF**

Apart from identifying the last row number programmatically, it may be possible that an excel file may contain empty undefined rows in the end.

So, the UploadParser also defines a default integer value :

public int EMPTYROW\_THRESHOLD = 50;

i.e after 50 null rows, the parser would itself imagine EOF and thus stop parsing.

This value can be altered by the user. Before calling the function to parse the uploaded file, add the following line of code :

triggerParse.EMPTYROW\_THRESHOLD = 10;

Thus the final code to call the plugin would be :

using UploadParser;

...

// Funtion in main program to call the plugin

public string parseUploadedFie(Stream xlsUpload)

{

...

TriggerParse triggerParse = new TriggerParse();

public int EMPTYROW\_THRESHOLD = 50;

Object File = triggerParse.parseFile(fileUpload,true, @"C:\\MyDir\");

...

}

1. **Default string to be replaced with “\n”**In UploadParser, it is assumed that a row terminates with “\r\n”.  
   But there may exist entries which contain “\n”, and thus may show up in excel file as a new line, even though it is a part of the same row entry.  
   To handle such issue, Upload Parser identifies the special character : “\n” and by default replaces it with “\\n”.

contents = Regex.Replace(contents, "(?<!\r)\n", "");