# Importer Framework

This framework is to be used to import CSV files into a database table.

## Structure of the Project

The project is composed for two main projects, attached to a project group named “ImporterPG.groupproj”. After open this project it will be available two projects, as follows:

1. Importer\ImporterApp\ImporterApp.dproj – an example application in how to use the framework
2. Importer\ImporterTest\ImporterTest.dproj – test project of the framework. It was used DUnitX to the unit testing.

The Framework has a Unit Called “ImporterFrameworkU.pas” which has all the classes of the framework.

## The Import Process

The framework is able to import a CSV file of a given File Definition structure, which is defined as we can see later in this document.

It can import all the fields of the structure or only some of them. The way to perform it is described below:

1. To import all the fields on the structure (File Definition):
   1. The file itself must not to have any header in it, and the property HasHeader (TCSVSourceFile) must be marked as False. Then the framework will import all the fields of the structure. Actually, it will demand that the record has to have the same structure of the file structure.
2. To import a given list of the fields, which can be all of them or just some of them:
   1. The file must to have a header which correspond to the field names to be imported, and the property HasHeader (TCSVSourceFile) must to be marked as True. Hence, the framework will process and import only the fields within the header of the file to be imported.

There are three main components which are needed to run the process:

1. TCSVSourceFile
   1. The “Create” expects a parameter
      1. FileDefinition file name which is the file that defines the file to be imported, as described later in this document;
   2. A property “FileName” must to be filled with the File which will be imported;
   3. By default, the TCSVSourceFile has the RecordDelimiter defined as a ‘;’ the TextDelimiter defined as ‘”’ and the FieldDelimiter defined as ‘,’. It can be changed to something else, but it will affect the reading of the FileDefinition and the File to be imported.
2. TDatabaseDestination
   1. The “Create” expects two parameters:
      1. TFDScript SQL component connected into a database;
      2. The TableName target of the import
3. TImportEngine: this is the responsible for the process itself
   1. The “Create” expects the following parameters:
      1. A TCSVSourceFile
      2. A TDatabaseDestination
      3. And a TProgressBar: this parameter is not mandatory, but the the framework is able to control it showing the progression of the process.

In order to expand the Source to the project an extended class can be developed based on the class “TSourceFile”. There is already a skeleton of a TFlatSourceFile, which need some work.

## The File Definition

The framework is able to create and manage CSV files, based in a given structure which must to follow the definition below:

The file definition must to be a Header with the names of each column defining the structure of the file to be imported to the database table.

### File Definition Structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Meaning** | **Mandatory** | **Content Expected** |
| FIELD\_NAME | The name of the field to be imported. It must be the same name on the header o the file to be imported | Yes | A unique String name |
| DATA\_TYPE | The type of data expected on this field while importing | Yes | It expects the following list of datatypes:  String  Currency  Integer  Numeric  DateTime  Boolean |
| ACCEPT\_EMPTY | This determines whether the framework will validate the content of the field. If True, the framework will accept this field empty. | Yes | Expects  True  False |
| VALIDATE\_LENGTH | Defines whether the framework will validate the length of the field.  Applicable only to the datatype String. | Yes | Expects  True  False |
| MAX\_LENGTH | Defines the max length of a given String. | If the “VALIDATE\_LENGTH” is True, then Yes | Expects a positive integer. |
| DESTINATION\_FIELD | The name of the target field on the database table. Applicable only when the target field has a different name given on the field “FIELD\_NAME” | Not | Expects a String name |

### Validations of the Importer

Based on the definition above, the “engine” is able to validate and reject if necessary the following datatypes:

* String
  + Empty or not, length and if it is an actual string
* Currency
  + Empty or not and if it can be converted to a Currency
* Integer
  + Empty or not and if it can be converted to an Integer
* Numeric
  + Empty or not and if it can be converted to a Numeric
* DateTime
  + Empty or not and if it can be converted to a DateTime
* Boolean
  + Empty or not and if it can be converted to a Boolean

### File definition example file

FIELD\_NAME,DATA\_TYPE,ACCEPT\_EMPTY,VALIDATE\_LENGTH,MAX\_LENGTH,DESTINATION\_FIELD;

Make,String,FALSE,True,50,;

Model,String,FALSE,True,60,;

Base Price,Currency,True,False,,BasePrice;

Cylinders,Integer,True,False,,;

Power,Integer,True,False,,;

Fuel,String, True,True,20,;

Weight,Numeric,True,False,,;

Introduced,DateTime,True,False,,;

Convertible,Boolean,True,False,,IsConvertible;

## Things to be implemented

The framework does not permit configuration to do the insertion in batches of a given number of valid records.

The framework can be expanded through implementing a child of the class TFileDefinition, to receive the FileDefinition of a recordset from a Database table.

At the moment, the framework only works with SQL Server as RDBMS. Other versions can be implemented, but some extra effort will be needed.

It is possible to increase the number of test cases. So far the cover the majority of the key processes inside of the framework.

In order to make the tests pass in another environment, some work will be needed on the given constants inside of the framework unit:

1. CSVFileDefinitionInvalid, pointing to the file “CSVFileDefinitionInvalid.csv”
2. CSVFileDefinitionValid, pointing to the file “CSVFileDefinitionValid.csv”
3. CSVSourceFileValid, pointing to the file “SourceFileValid.csv”
4. CSVSourceFileInvalid, pointing to the file “SourceFileInvalid.csv”

All the mentioned above files are on the main folder of the project group, inside of a subfolder named “SampleFiles”.

## Conclusion

This framework can be adapted and used and changed as needed.

Thank you!