# Usage of Excel import/export for Eclipse/Capella (EMFSpreadsheetIO)

## Installation

The component is given as an update site : a set of eclipse plugin with a sample importer able to import a list of EClass into an ePackage from an ecore file.

There is a mandatory dependency : Apache POI. This component is available in the m2doc update site (<https://github.com/ObeoNetwork/M2Doc> ), because m2doc, the Obeo document generation solution, is also based on the same version of POI.

Engine plugins must be installed using the Help->Install Software installation command from Eclipse or Capella. As the engine is based on EMF, it can be installed in Capella or every EMF based Modeler.

Plugins contents

|  |  |
| --- | --- |
| org.obeonetwork.spreadsheet.importer | Excel Import Engine |
| org.obeonetwork.spreadsheet.exporter | Excel Export Engine |
| org.obeonetwork.spreadsheet.importer.ui | Graphical User Interface to launch an import into a model, and the export of a part of a model. |
| org.obeonetwork spreadsheet.exporter.wizard | Codeless wizard to export all instances of a Metaclass, with its attributes. |
| org.obeonetwork.spreadsheet.importer.automatic | Automatic importer based on the previous exported, with reconciliation guessing. |
| org.obeonetwork.spreadsheet.importer.ecore.example | Sample importer based on EMF, to import a set of EClass into an ePackage. |
| org.obeonetwork.spreadsheet.importer.capella.example | Basic importer based on Capella |
| com.thalesgroup.clarity.excelimport.examples | Capella Sample importer presenting the import of hierarchical functions, and the link to Requirements. |

## Importer : User usage

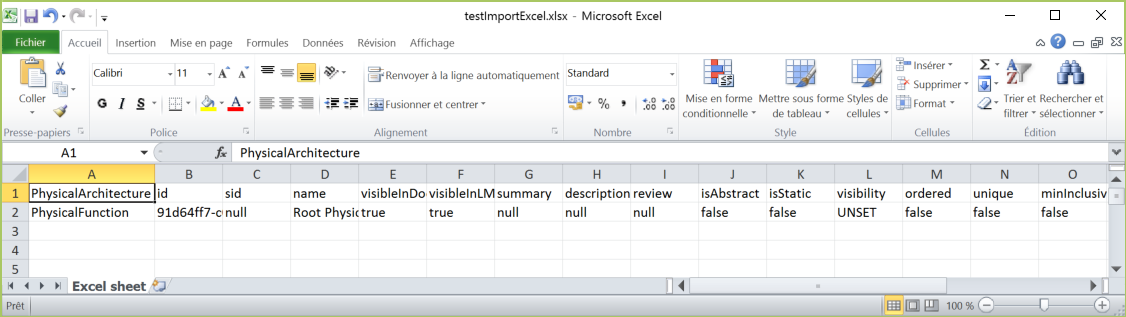
The importer interface is located into a single command. In its explorer (Model Explorer or Capella Explorer), the user selects an xlsx file and its import destination. The, he uses right click, and then « import » in the correct category.

|  |  |
| --- | --- |
| Here is an excel source : a list of eClass with, for each one, its abstractness, inheritance, and an attribute name. |  |
| We launch the import with excel selection in the explorer, and the destination, here, and an ePackage. |  |
| A dialog box open next, to ask the user the importer to use. We choose the ecore importer and then click Ok. |  |
| The excel file is correctly imported. |  |
|  |  |

## Exporter : Usage by User

Excel exporter is available in the context menu. From an EMF model element, you select the command « Export ».

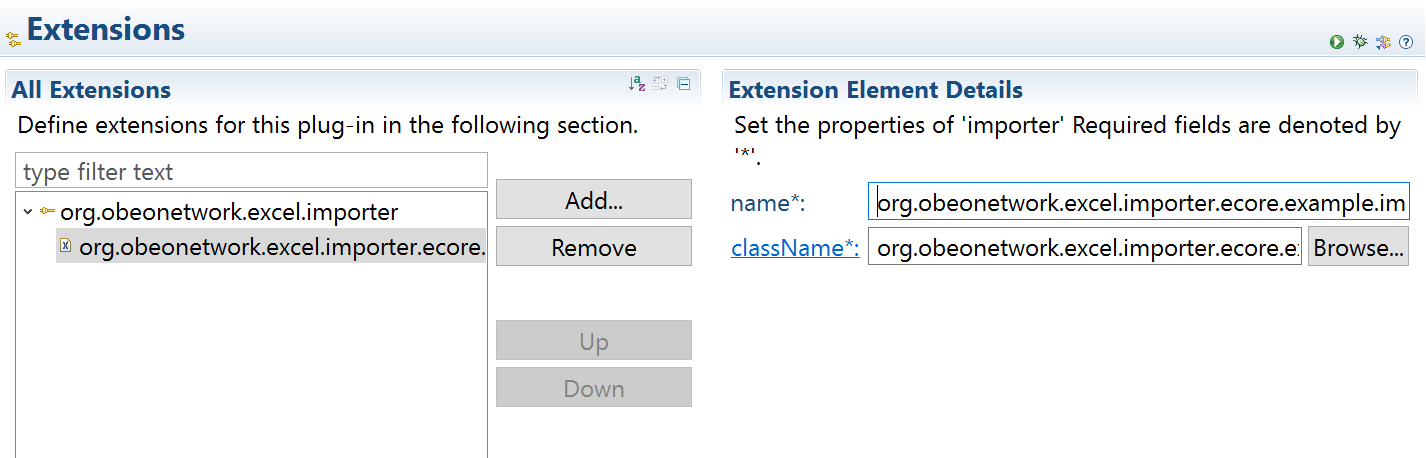
|  |  |
| --- | --- |
|  | Excel exporter is available in the context menu. From an EMF model element, you select the command « Export ». |
| A box opens to choose the exporter to use. By default, the exporter wizard is selected, but the user can select another, which cannot have a graphical user interface. |  |
|  | This dialog gives the user the possibility to select a metaclass. Then, the engine will export all instances of this metaclass in the model subtree, starting from the context object. |
|  |  |



And this is the exported file. One line per exported object, and columns corresponding to its attributes. We can also notice all extensions and Property Values from these elements.

## Create a new Importer

The creation of a new importer is done in java, by implementing an extension point published by the import engine. This chapter will explain the different steps to create a new import format.

You have to start by creating a new plugin project, and in the dependencies, add the engine plugin, org.obeonetwork.spreadsheet.importer. Then, in the extensions tab, you have to implement the extension point with the same name.

In this extension point, the className attribute is a plugin class that implements the org.obeonetwork.spreadsheet.importer.IExcelImporter interface. It’s the implementation of this interface which will make it possible to specify the behavior of the importer.

The principle of the importer is a path by line of the excel file, and for each line, an action is called on the interface, passing as a parameter the list of contents of cells, empty cells included not to lose data alignments.

There are 5 five methods to implements:

|  |  |
| --- | --- |
| public String getName() | Returns the name of the importer, it is displayed in the import dialog |
| public void setDestination(EObject destinationObject) | Receives the EMF destination of the import; This method allows you to keep it in an attribute of the class, but also to verify that the importer is called on the correct type of object. |
| public void computeFirstLine(List<String> lineData) | Called when parsing the first excel line with content. This method often makes it possible to recover the information of abscissa, which can be variable. It can also be used to verify that it is indeed the expected Excel format. |
| public void computeOtherLine(int lineNumber, List<String> lineData) | Central method of the import, this method is called for each line of the file by passing the data of the line, possibly with the empty chains for the empty cells, and the number of the line (starting with 0, watch out for the difference with Excel) |
| public void importEnded() | Method called at the end of the import. It is often the moment to validate the model obtained, even to create links between the imported elements, because it is the moment when the whole file has been mounted in memory. |

The example of import ecore has been specially created to help understanding. Do not hesitate to refer to it to start your own importer.