# 0 — Quick start

When I get my hands on a new piece of software (be it a game or any other application) I cannot wait to start using it, without ‘wasting’ too much time reading the instruction manual.

This introductory chapter is dedicated to all those who share this attitude.

In it, I will try to give only the essential elements for a test ‘ride’ with the plugin, assuming that the reader has at least a medium knowledge of Python environment, Sigil and WinWord.

You will find all the boring stuff in the following chapters.

**WinWord**

To run this plugin you need to have WinWord on your PC.

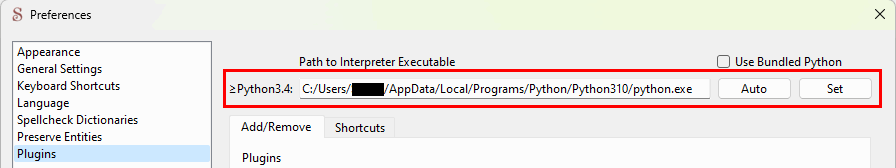
**Python environment**

Following Python extensions are required:

* PyQt5
* pywin32
* QScintilla

**Sigil setup**

1. From Sigil’s menu choose Plugins → Manage Plugins and set the Python interpreter path.



Picture 0-1

1. Download the docxTranslat.zip plugin
2. Add the plugin to Sigil

**Test file**

The plugin works properly only for DOCX documents formatted according to a few simple rules (see § 2).

For a first test, I recommend downloading the **docxTranslat.docx** file onto your PC.

**Plugin docxTranslat setup**

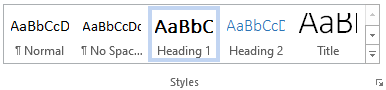
1. Run the plugin from Sigil.
2. In the ‘Config’ tab, check the values in the fields ‘WinWord exe path’, ‘Editor exe Path’, ‘Working dir Path’. Change them according to your PC configuration using the buttons ‘set WinWord exe’, ‘set Editor exe’, ‘set Working folder’. All these fields are mandatory and the plugin will check for their existence before to start the process of importing a file into Sigil.

Tips

You can use Notepad.exe as an editor.

I recommend creating an empty working folder so, at the ending of import process, you can check easily for the files created by the plugin.

1. Open **docxTranslat.docx**
2. Move the mouse on a chapter heading of docxTranslat.docx (e.g. **0 — QuickStart**)
3. Check for the active style in WinWord ‘HOME’ tab



Picture 0-2

1. If the value in the ‘doc split style’ field is different from the name of the active style, set the value of the field to the name of the active style (e.g. Heading 1)
2. Save the configuration values by clicking on the ‘save settings’ button
3. Close **docxTranslat.docx** and all open WinWord documents.
4. Click on ‘choose .docx’ button (bottom left in the tab)
5. Choose docxTranslat.docx from the folder where you downloaded it.
6. Click on the ‘create .csv’ button to start the analysis of docxTranslat.docx file
7. At the end of the analysis, go to the ‘Translation’ tab
8. Click on the ‘choose .docx’ button
9. Choose docxTranslat.docx from the folder where you downloaded it.
10. The plugin fills a grid with the results of the previous analysis. Each line in the grid corresponds to a chapter in docxTranslat.docx file. Some fields in the tab (‘File HTM’, ‘Word doc start page’, ‘Word doc end page’) are still empty
11. Choose a line (i.e. a document chapter) of the grid by clicking on it
12. The plugin fills the tab’s empty fields
13. Click on the ‘start translation’ button and wait for the message in the log area at the bottom of the page (…TRANSLATION PROCESS SUCCESSFULLY COMPLETED…)
14. Click on the ‘load into Sigil’ button and wait for the message in the log area
15. Quit the plugin by clicking on the ‘exit’ button and see the loaded file from Sigil interface.

# 1 — Requirements

**SOFTWARE**

* Windows 10- 11
* Microsoft Office 2010- 2013
* Python 3.6.5- 3.10.2
* Sigil 1.9.30- 2.0.2

Python extensions:

* PyQt5 5.15.6 (GUI)
* pywin32 304 (access to Windows API)
* QScintilla 2.13.3 (plugin mini editor)

**Tips to install additional Python extensions.**

* From Windows command prompt (cmd) go to your Python installation folder ( e.g. cd C:\Users\your\_user\AppData \Local\Programs\Python\Python310)
* Go to ‘Scripts’ folder
* From the Scripts folder install the required modules running following commands:
  + pip install pyqt5
  + pip install qscintilla
  + pip install pywin32

**TESTED ON**

1. Clone PC (development system):

* Processor: 11th Gen Intel(R) Core(TM) i5-11400 @ 2.60GHz 2.59 GHz
* RAM: 32 GB
* Windows 11, 64-bit operating system, x64-based processor
* Microsoft Office 2013
* Python 3.10.2
* Sigil 2.0.2

1. Hp Pavilion

* Processor: Intel(R) Core(TM)2 Duo CPU T6600 @ 2.20GHz 2.20 GHz
* RAM: 4 GB
* Windows 10, 64-bit operating system, x64-based processor
* Microsoft Office 2013
* Python 3.6.5
* Sigil 1.9.30

1. Acer laptop Aspire E5—573G

* Processor: Intel(R) Core(TM)i5—4210U CPU @ 1.70GHz 2.40 GHz
* RAM: 8 GB
* Windows 10, 64-bit operating system, x64-based processor
* Microsoft Office 2010
* Python 3.6.5
* Sigil 2.0.1

# 2 — Basic info

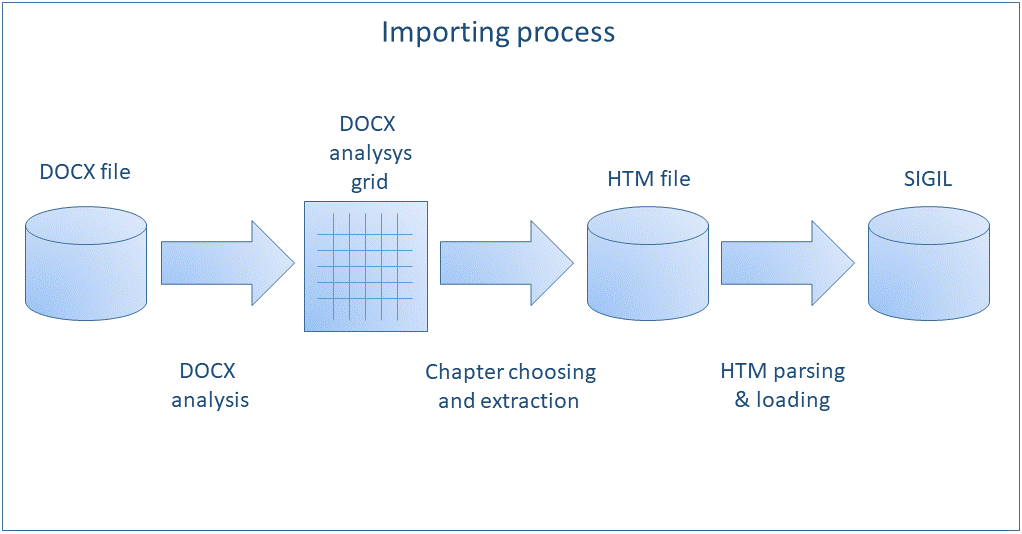
**Plugin overview**

The plugin name is **docxTranslat**.

It requires a WINWORD.EXE installation to work properly and it allows the user to import DOCX documents into Sigil chapter by chapter. It imports text, lists, tables and images preserving most of the formatting (bold, italic, colors and so on) used in the DOCX document.

The process of importing a single chapter into Sigil has the following steps:

1. Analysis of the whole DOCX document: **docxTranslat** requires an analysis of the DOCX document before to import its chapters into Sigil. The plugin analysis tool reads the DOCX document, splits it (virtually) at each chapter heading and stores the results into a CSV file.
2. The user can display the CSV file content in a grid, each line of which corresponds to a chapter.
3. The user chooses one specific line (i.e. a chapter) from the grid and starts the importing process.
4. The plugin extracts from the DOCX document the chosen chapter, saves it in a working folder as a HTM (web-filtered page) file and parses it to obtain a file suited for Sigil
5. The user can preview and modify (not recommended!) the file before loading it into Sigil.



Picture 1—1

The plugin provides also some other tools to:

* import images (jpg, png, gif) into Sigil.
* create in Sigil a text file starting from an imported image.
* view Sigil Spine files, modify their order and delete unwanted files.
* view Sigil Manifest files.
* create specific XHTML pages for epub (Title, Half Title, Copyright, Dedication, Epigraph and Table Of Content or TOC).
* execute Python scripts in order to interact with the Sigil BookContainer class (bk)

# 3 — On DOCX files formatting

For the plugin to work properly, the DOCX document must follow some simple rules:

* The document must have at least one chapter.
* All chapters must start with a heading having the format

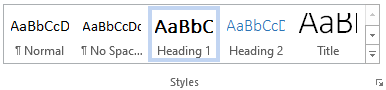
**xx dash TITLE**

Heading example: 1 — Requirements

The plugin will translate the headings as follows:

|  |  |
| --- | --- |
| DOCX document | Sigil |
| **1 — Requirements** | **1**  **Requirements** |

* The dash must be one of the following: minus sign (e.g. from the numeric keyboard, chr(45)) or ‘double minus’ (i.e. —, chr(8212)).
* No more than one heading on the same page allowed
* All headings must have the same WinWord style, e.g.:



* The style name is a part of the plugin configuration (see § 5.1.1 — Plugin configuration. Basic settings).
* No HTM code allowed into DOCX chapters
* No angle brackets  (chr(60), chr(62)) allowed

**Additional note on DOCX document**

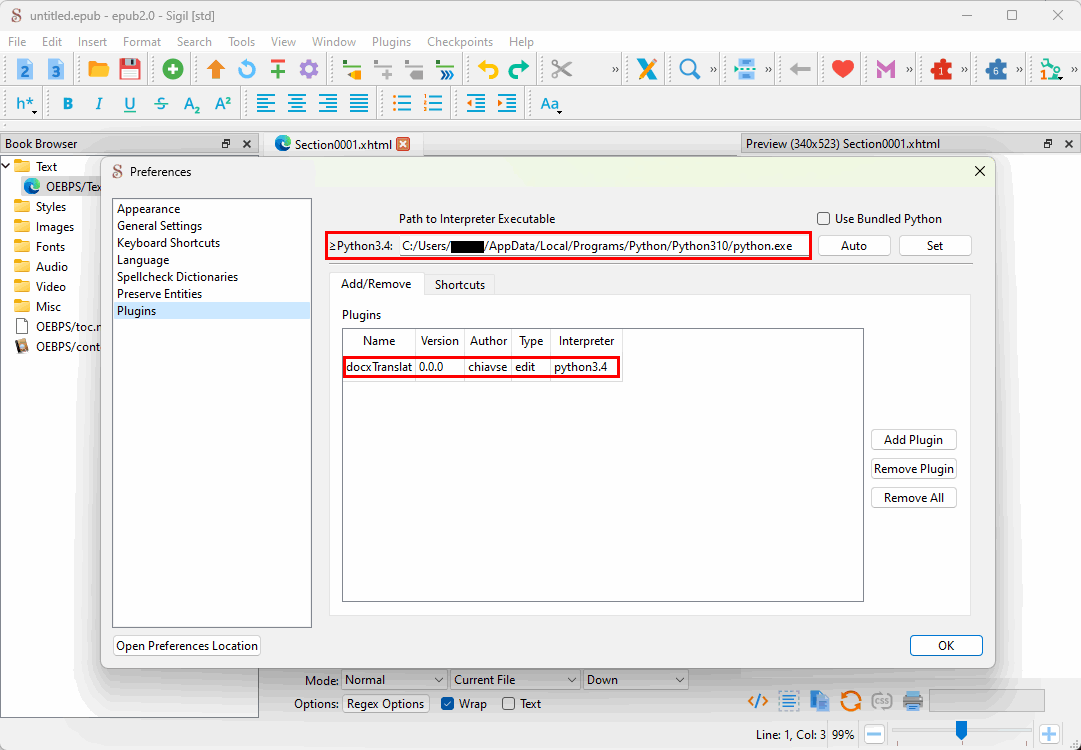
* Each chapter may contain one or more images. The plugin will import them automatically into Sigil.
* The plugin does not handle footnotes.

# 4 — Installing and starting

Before to start be sure having all required software modules installed on your Windows PC.

Installing steps:

1. Download docxTranslat.zip
2. Start Sigil
3. From Sigil menu bar select: Plugins → Manage Plugins
4. Uncheck ‘Use Bundled Python’ if checked
5. Using the ‘Set’ button choose Python interpreter path
6. Using the ‘Add Plugin’ button load docxTranslat.zip into Sigil
7. After loading docxTranslat.zip, the Manage Plugins page will see as in the following picture:



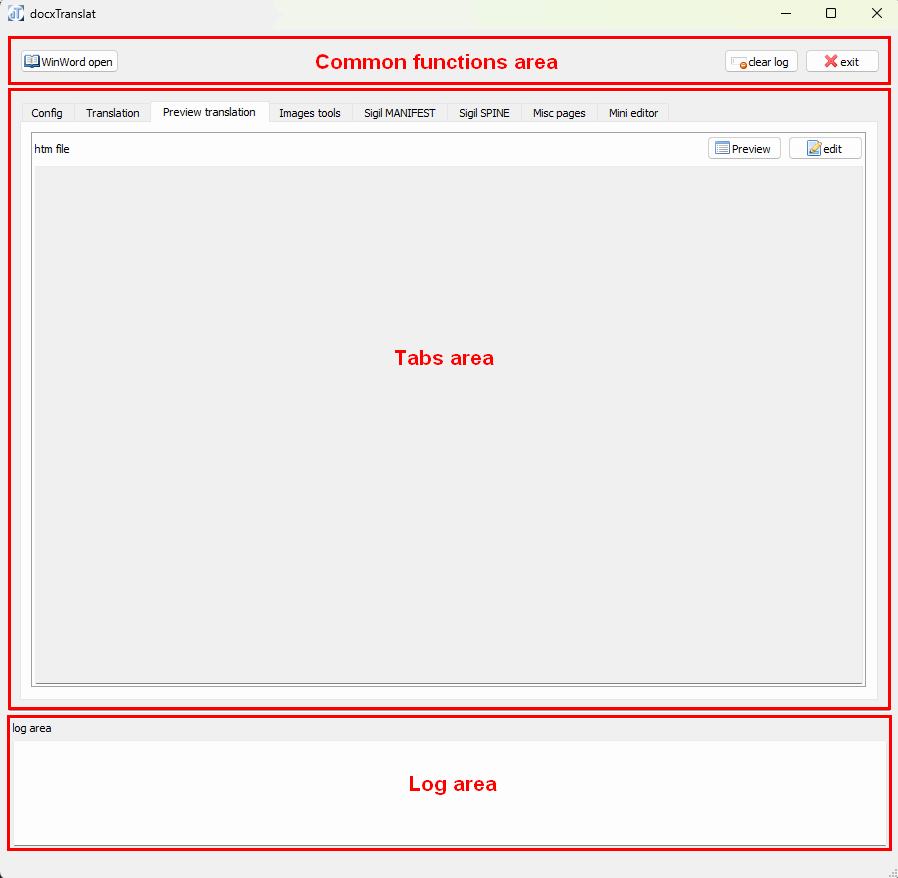
Picture 4—1

1. Starting docxTranslat: from Sigil menu bar select: Plugins → Edit → docxTranslat

# 5 — docxTranslat user interface

docxTranslat user interface has three different areas:

1. Common functions area. It gives access some functions available to the user whatever the chosen tab.
2. Log area to display messages/ errors
3. Tab area. Each tab gives access to specific sets of docxTranslate functions



Picture 5—1

The user may change the size of ‘Tabs area’ and ‘Log area’ by dragging the boundaries between the two areas.

**Common functions area user interface**:

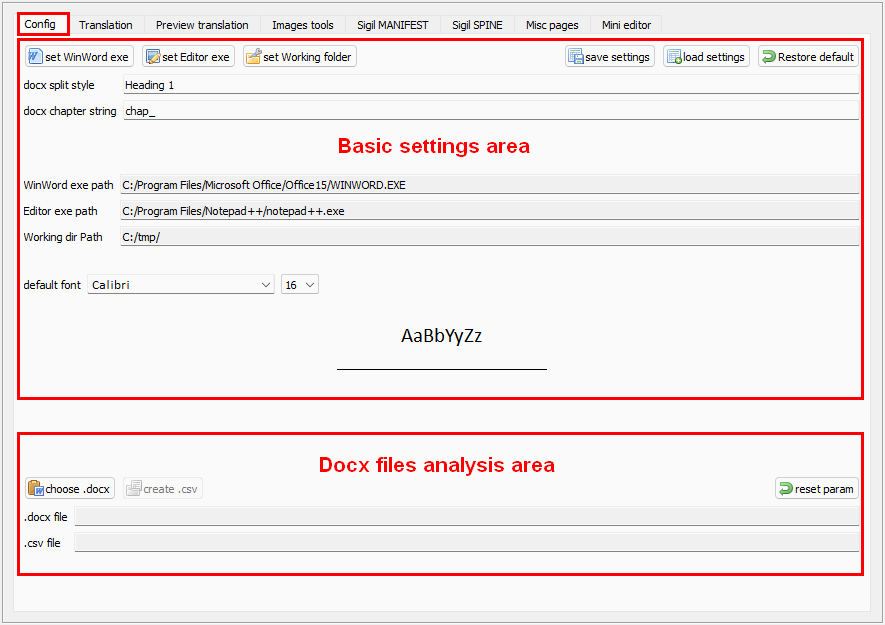
|  |  |
| --- | --- |
|  | Opens a dialog that allows the user to choose and open a DOCX document |
|  | Clears the log area |
|  | Quit the plugin |

# 5.1 — Plugin configuration

To access plugin configuration click on tab ‘Config’.

From this tab the user can:

1. Access, change and save the plugin basic settings (‘Basic settings area’)
2. Analyze DOCX files by virtually splitting them into chapters before to import them into Sigil (‘DOCX files analysis area’)



Picture 5.1—1

# 5.1.1 — Plugin configuration. Basic settings

To work properly, the plugin requires the setting of some parameters.

On the first run, the plugin reads the default values of the parameters basic settings from the hConstant.py module. The user can change the values of the parameters basic settings and save them into a configuration file. At each subsequent execution, the plugin will load the basic settings from the configuration file.

The configuration file has name **docxTranslat.txt**.

The plugin saves and loads the configuration file into the following folder:

C:\Users\your\_user\AppData\Local\sigil-ebook\sigil\plugins\docxTranslat\docxTranslat.txt

Required parameters list below.

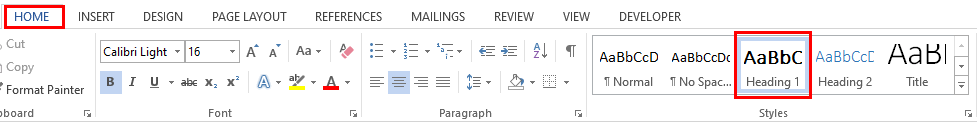
**— ‘docx split style’ field**.

Default value: ‘Heading 1’.

The plugin provides a tool to read the DOCX file and to split it (virtually) at each heading having the name specified by this parameter (see § 3 and § 5.1.2).

The user must set the value of this parameter according to the style names in WinWord ‘HOME’ tab.

The user can change the default value by editing the field.



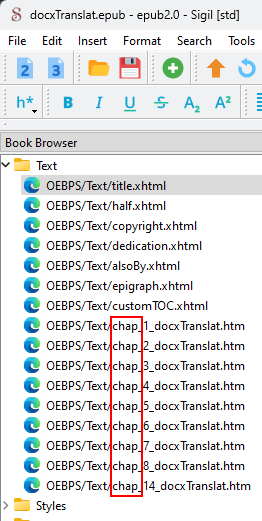
Picture 5.1.1—1

**— ‘docx chapter string’ field**.

Default value: chap\_

This parameter is the prefix of the name for the text files loaded into Sigil.

The user can change the default value by editing the field.



Picture 5.1.1—2

**— ‘WinWord exe path’ field**.

Default value: C:/Program Files/Microsoft Office/Office15/WINWORD.EXE

This parameter is the path of WinWord executable. The plugin uses this parameter to start WinWord and open a DOCX file.

The user can change this parameter by clicking on the ‘set WinWord exe’ button.

**— ‘Editor exe path’ field**.

Default value: C:/Program Files/Notepad++/notepad++.exe

The plugin uses this parameter to activate an external editor in order to preview/ change (not recommended!) the HTM files before loading them into Sigil.

The user can change this parameter by clicking on the ‘set Editor exe’ button.

**— ‘Working folder’ field**.

Default value: C:/tmp/

This parameter is the path used by the plugin in order to:

* write a .CSV file with the results of the DOCX file analysis (see § 5.1.2);
* write the HTM files extracted from DOCX file. If a HTM file contains one or more images, the extraction process will create a sub—directory where to save all images. If the extracted file has name chap\_n\_docxTranslat.htm, the sub—folder will have the name:

chap\_n\_ docxTranslat \_files;

Please note: if in WinWord FILE → Options → Language → Choose Display and Help Language you set Italian [Italiano] as default, the images sub—folder name will be

chap\_n\_ docTranslat \_file instead of chap\_n\_ docTranslat \_file**s**.

The plugin works with both Italian and English settings. No other languages tested.

* rewrite the HTM file after parsing it to make it suitable for uploading into Sigil.

**— ‘default font’ fields.**

These parameters are the font family and the dimension of the font characters.

The user can change the parameters by selecting different values from the combo boxes.

**Basic settings user interface**:

|  |  |
| --- | --- |
|  | Opens a dialog that allows the user to choose a different path for WinWord. |
|  | Opens a dialog that allows the user to choose a different path for the external editor. |
|  | Opens a dialog that allows the user to choose a different path for the working directory. |
|  | Displays the list of the installed fonts to allow the user to change the default font by choosing a new one. |
|  | Displays the list of allowed character dimensions to allow the user to choose a new one. |
|  | Displays an example of the selected font family/ characters dimension |
|  | Reads basic settings values from the page and save them into the file docxTranslat.txt |
|  | Loads basic settings values from the file docxTranslat.txt and load them into the page fields |
|  | Restores the initial values of the basic settings by reading them from hConstant.py module and loading them into page fields |

**Basic settings in hConstant.py module**

winWordExePath= ‘C:/Program Files/Microsoft Office/Office15/WINWORD.EXE’

editorExePath= ‘C:/Program Files/Notepad++/notepad++.exe’

workingDir= ‘C:/tmp/’

#

CSV\_STYLE\_TO\_SEARCH= 'Heading 1'

CHAPTER= 'chap\_'

# default font family

defaultFontFamily= 'Calibri'

defaultFontSize= '16'

**Basic settings in file docxTranslat.txt**

winWordExePath=C:/Program Files/Microsoft Office/Office15/WINWORD.EXE

editorExePath=C:/Program Files/Notepad++/notepad++.exe

workingDir=C:/tmp/

docxSplitStyle= Heading 1

docxNameChapStr=chap\_

fontFamily=Calibri

fontSize=16

# 5.1.2 — Plugin configuration. Docx file analysis

The user must analyze each DOCX file before to start the importing of its chapters into Sigil.

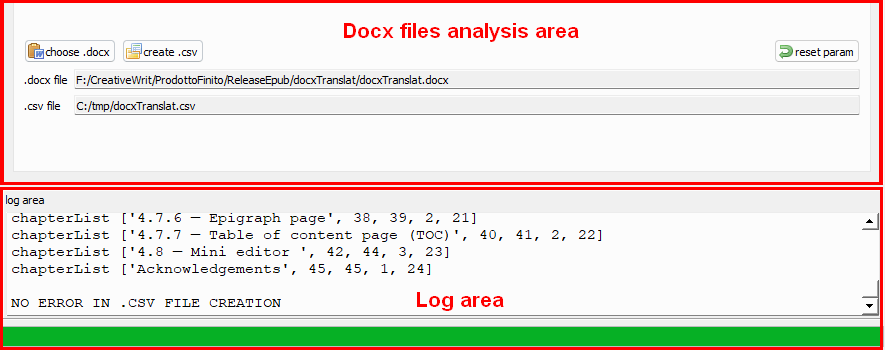
The analysis starts by choosing and opening a single file to get the total number of pages and paragraphs contained in the file.

The following step is the scan of each page, paragraph by paragraph.

A page is marked as the first page of a chapter if it contains a paragraph written with the ‘docx split style’ defined in the basic settings.

The plugin records the scan results in a CSV file written in the ‘Working directory path’ defined in the basic settings. The CSV file uses “;” as separator, has the same name of the just analyzed DOCX file and, obviously, CSV extension.

A progress bar at the bottom of the page and messages in the log area allow the user to follow the file analysis process. At the end of the file analysis, the ‘Docx files analysis area’ and the ‘Log area’ will look as in the following picture.



Picture 5.1.2—1

**Be sure to repeat the analysis every time you add a new chapter or change the length of an existing one.**

Docx file analysis user interface:

|  |  |
| --- | --- |
|  | Opens a dialog that allows the user to choose a DOCX file.  Writes DOCX file full path in ‘.docx file’ field.  Writes CSV file full path in ‘.csv file’ field. |
|  | Enabled after choosing a DOCX file to analyze.  Starts the DOCX file analysis process. |
|  | Deletes the content of ‘.docx file’ and ‘.csv file’ fields.  Disables ‘create .csv’ button. |
|  | Deletes progress bar and clears log area  (the button is in the Common function area) |

**CSV file fields**

* chapter title
* start page number
* end page number
* number of pages in chapter (only info, not used in the following elaboration)
* output prefix

**CSV example**

The analysis this document generates a CSV file having the following content:

Title;startPage;endPage;pageNr;output prefix

0 — Quick start;1;2;2;1

1 — Requirements;3;4;2;2

2 — Basic info;5;6;2;3

3 — On DOCX files formatting;7;7;1;4

4 — Installing and starting;8;8;1;5

5 — docxTranslat user interface;9;9;1;6

5.1 — Plugin configuration;10;10;1;7

5.1.1 — Plugin configuration. Basic settings;11;13;3;8

5.1.2 — Plugin configuration. Docx file analysis;14;15;2;9

5. 2 — Docx translation;16;19;4;10

5.3 — Preview translation;20;21;2;11

5.4 — Images tools;22;23;2;12

5.5 — Sigil MANIFEST;24;24;1;13

5.6 — Sigil SPINE;25;26;2;14

5.7 — Misc pages;27;30;4;15

5.7.1 — Title page;31;32;2;16

5.7.2 — Half title page;33;33;1;17

5.7.3 — Copyright page;34;34;1;18

5.7.4 — Dedication page;35;35;1;19

5.7.5 — Other titles by the same author;36;36;1;20

5.7.6 — Epigraph page;37;37;1;21

5.7.7 — Table of content page (TOC);38;39;2;22

5.8 — Mini editor ;40;41;2;23

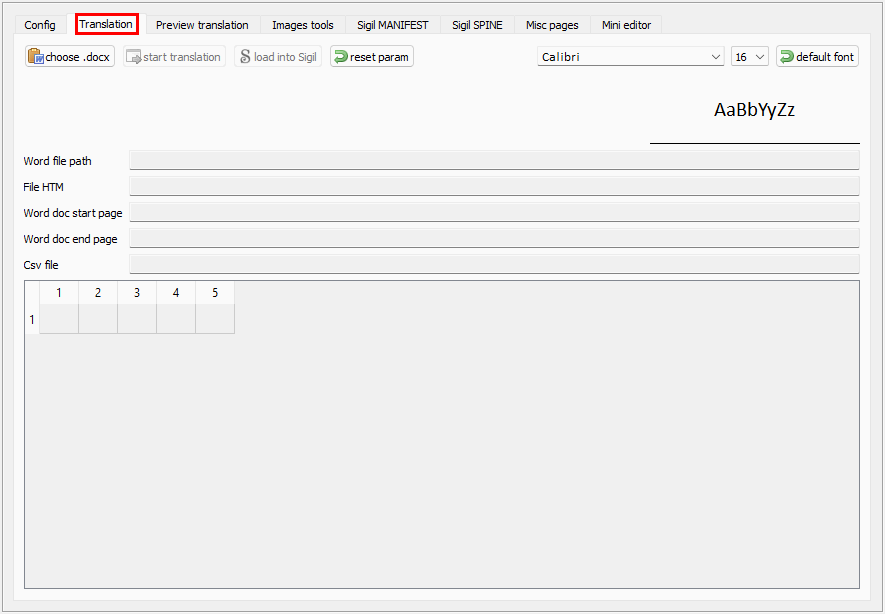
Acknowledgements;42;42;1;24

# 5. 2 — Docx translation

**Be sure to close all the opened docx files before to start the translation of a docx file.**

The user begins the translation process by choosing the ‘Translation’ tab.

At the beginning of the process, the Translation tab will look as in the following picture.



Picture 5.2—1

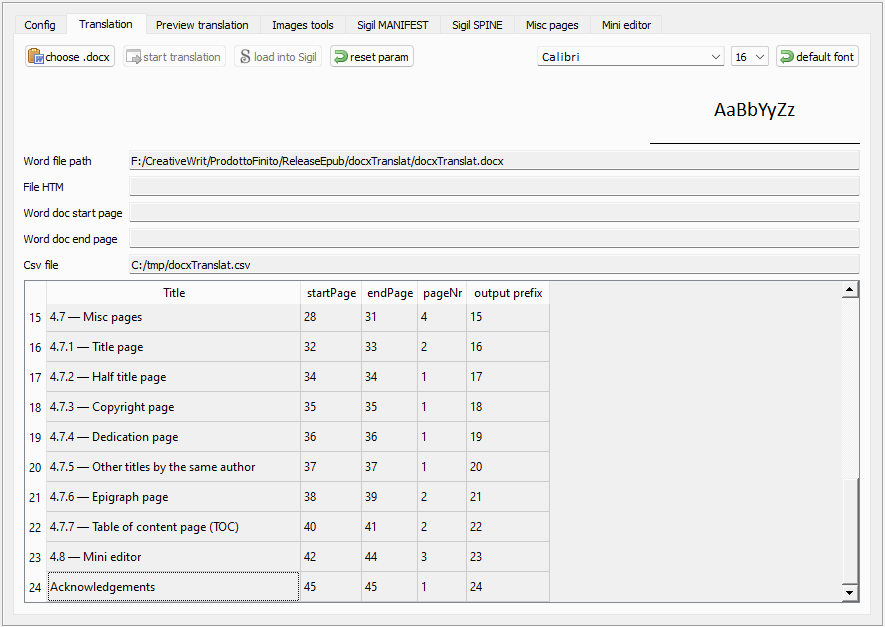
The user has to choose the DOCX file to translate by clicking on the ‘choose .docx’ button.

If the user has not yet analyzed the chosen DOCX file (i.e. the corresponding CSV file does not exists in the Working folder), the plugin displays an error message in the log area.

If the user has already analyzed the chosen DOCX file (i.e. the corresponding CSV file exists in the working folder), the plugin fills:

* the field ‘Word file path’,
* the field ‘Csv File’,
* the grid, reading the values from the CSV file,

The ‘Translation’ tab will look as in the following picture:



Picture 5.2—2

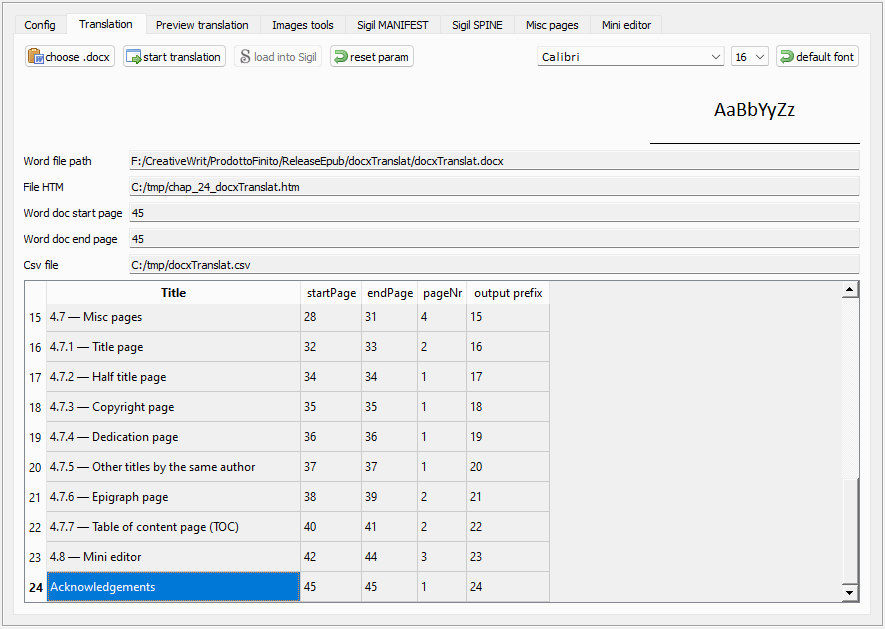
To continue, the user must choose a line (corresponding to a chapter of the DOCX file), by clicking on a line of the grid.

The plugin will fill:

* the field ‘file HTM’ (name of the file to be loaded into Sigil). If the DOCX file to import into Sigil has name docxTranslat.docx (as in the previous picture), each HTM file will have name **start\_n\_docxTranslat.htm**, being:
  + ‘start\_ ‘ the value in basic settings filed ‘docx chapter string’
  + n the value of the grid cell of the ‘output prefix’ column corresponding to the selected line (chapter)
* the field ‘Word doc start page’ (chapter start page number)
* the field ‘ Word doc end page’ (chapter end page button)

and will enable the button ‘start translation’.

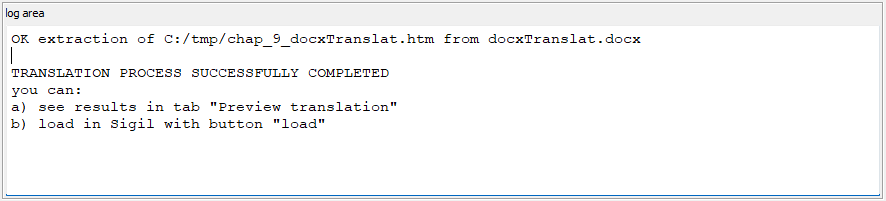
The ‘Translation’ tab will look as in the following picture:



Picture 5.2—3

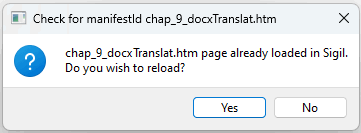
Before to start the translation the operator may change the family font and its dimension in points or restore default font settings as defined in the ‘Config’ tab.

If an error occurs during the translation it will be displayed in the ‘log area’, otherwise the plugin will enable the button ‘load in Sigil’ and will display in the ‘log area’ a message like the following:



Picture 5.2—4

If the operator tries to load a chapter already existing in Sigil, the plugin will display a message like the following:



Picture 5.2—5

User interface:

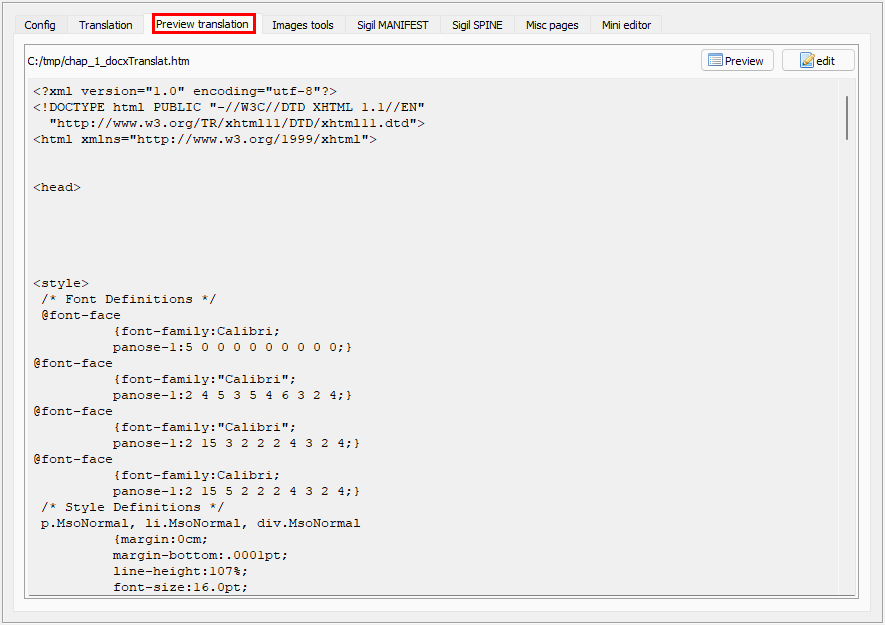
|  |  |
| --- | --- |
|  | Opens a dialog window to allow the user to choose the DOCX file to load into Sigil. |
|  | Enabled after the user chooses a line in the grid.  The user, by clicking on it, starts the translation process.  The plugin:   * Extracts the required chapter from DOCX file and saves the chapter into the Working folder as a HTM (web filtered) file * Parses the extracted HTM file in order to obtain a file suited for Sigil   If the translation process is ok, enables the ‘load into Sigil’ button and disables itself |
|  | Loads the parsed HTM file into Sigil.  If the loading is ok, disables ‘start translation’ button and itself. |
|  | Resets the page to its initial state (picture 5.2—1) :   * clears all the fields * clears the grid * disables the ‘start translation’ and ‘load in Sigil’ buttons. |
|  | Let the user choose the font family and its dimension in pts. |
|  | Displays a sample of the family font having the chosen dimension in pts. |
|  | Restores the default font settings as defined in the ‘Config’ tab |

# 5.3 — Preview translation

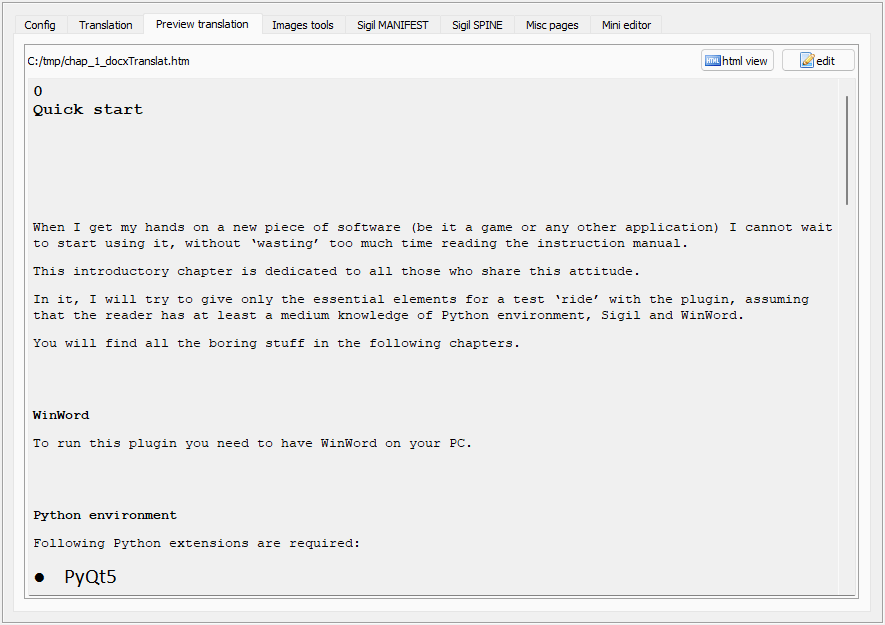
After the translation described in the previous chapter, the user may use the ‘translation’ tab to preview and modify (not recommended) the file which is about to load into Sigil.

By clicking on the ‘Preview’ button, it is possible to switch from the code view of the file (Picture 5.3—1) to a rough HTML view (Picture 5.3—2).

The user is also able to open/modify the file with the editor defined in the ‘Editor exe path’ field of the ‘Config’ tab.



Picture 5.3—1 Code view



Picture 5.3—2 HTM view

User interface:

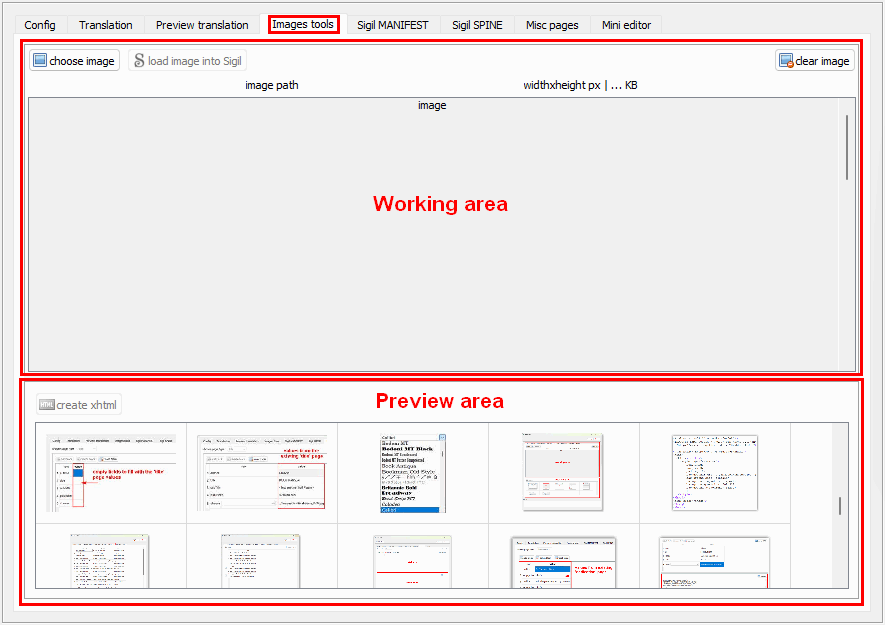
|  |  |
| --- | --- |
| Label at the top left of the ‘Tabs Area’ | Full path of the file that will be loaded into Sigil |
|  | Switches between code and HTML view of the file |
|  | Opens the file with the editor defined in the ‘Editor exe path’ field of the ‘Config’ tab. |

# 5.4 — Images tools

The ‘Images tools’ tab has two areas:

1. preview area,
2. working area.

The user may change the size of both by dragging the boundaries between the two areas.



Picture 5.4—1

With this tab, the user can:

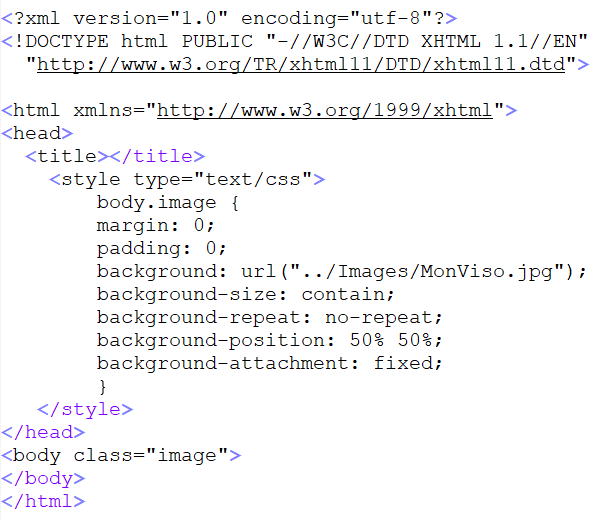
* view, in the ‘Preview area’, the thumbnails of the images (if any) already loaded into Sigil;
* load a new image into the Working area using the ‘choose image’ button;
* enlarge and shrink the image displayed in the ‘Preview area’ by using ctrl+ mouse wheel;
* load the image from the working area into Sigil using the button ‘load image’ (enabled after the operator loads and image into the Working area);
* clear the Working area by clicking on the ‘clear image’ button;
* load an image from the Preview area into the Working area by clicking on its thumbnail;
* create a text file using the image loaded into the working area and load the file into Sigil using the ‘create xhtml’ button (enabled after loading an image from the Preview area into the Working area).

User interface:

|  |  |
| --- | --- |
| ‘Image path’ label | Full path of the file loaded using ‘choose image’ button  or  name of the file loaded into ‘Working area’ from Preview area |
| ‘widthxheight px |… KB’  label | Dimension in pixels (width x height) and in Kb of the image displayed in the ‘Working area’ |
|  | Opens a dialog window that allows the operator to choose the image file to load into ‘Working area’.  Only jpg, png, gif files allowed.  Enables the ‘load image into Sigil’ button when the user chooses an image from the dialog box.  Sets the values of ‘image path label’ and of ‘widthxheight px |… KB label’.  Disabled when the user double clicks on an thumbnail in the ‘Preview area’ |
|  | Loads the image file into Sigil.  If the file already exists, the plugin displays a message asking the user if the reload is required. |
|  | Restores the page to its initial status, as in Picture 5.4—1 |
|  | Enabled after the user clicks on a thumbnail in the Preview area.  Creates a XHTML file using the image the user clicked on.  (see below example) |

Example.

Choosing the image MonViso.jpg from Preview area, the user can create in Sigil the file **IMG\_MonViso.xhtml** having the following code:



Picture 5.4—2

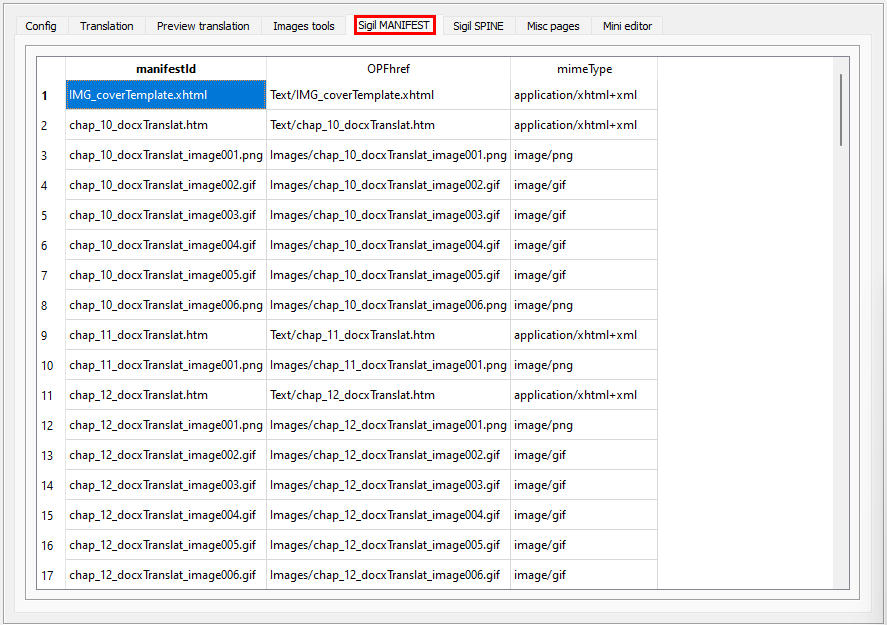
# 5.5 — Sigil MANIFEST

As for this documentation, suffice it to say that the MANIFEST is a section of the **content.opf** file which defines content and organization of the EPUB file created by Sigil. The MANIFEST contains the list of all the files that are part of the EPUB file (XHTML text files, images, stylesheets, etc.)

(Detailed information at: <https://idpf.org/epub/20/spec/OPF_2.0_latest.htm>).

From the 'Sigil MANIFEST' tab, the user can display the following info regarding the files listed in the Manifest section of the content.opf file:

* manifest\_id,
* OPF\_href,
* media-type.



Picture 5.5-1

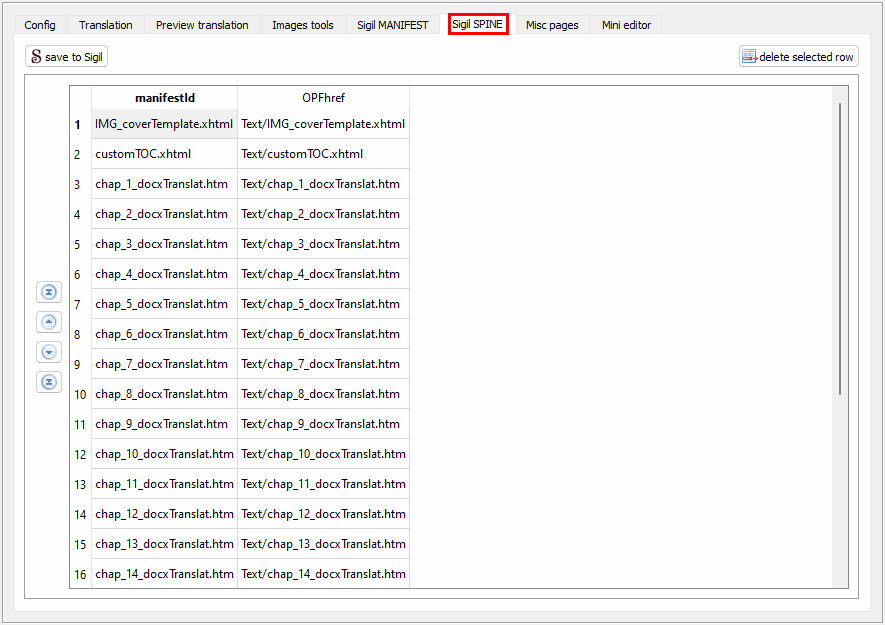
# 5.6 — Sigil SPINE

As for this documentation, suffice it to say that the SPINE is a section of the **content.opf** file which defines the content and organization of the EPUB file created by Sigil. The SPINE contains the list the text files (mime type= ‘application/xhtml+xml’) loaded into Sigil; their position in the list determines the order in which they appear when opening the EPUB file.

(Detailed information at: <https://idpf.org/epub/20/spec/OPF_2.0_latest.htm>)

From the 'Sigil SPINE tab, the user can display a grid with the following info regarding the files listed in the Spine section of the content.opf file:

* manifest\_id,
* OPF\_href.



Picture 5.6—1

The user can also:

* modify the order of the grid rows, changing the visualization order in the corresponding EPUB file;
* save any change into the SPINE section of the content.opf file;
* delete any row, but the last one, from the grid.

**User interface**:

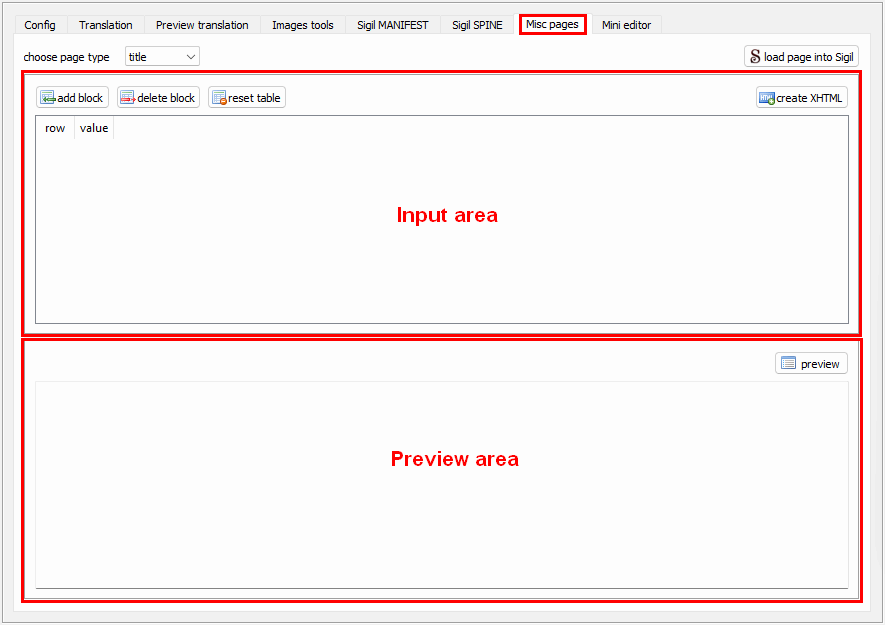
|  |  |
| --- | --- |
|  | Moves the selected row to the beginning of the grid |
|  | Moves up the selected row |
|  | Moves down the selected row |
|  | Moves the selected row to the end of the grid |
|  | Saves any change to the SPINE section of the content.opf file |
|  | Deletes the selected row.  The user must confirm the operation.  Deleting a row will delete the corresponding Sigil file and all its references from MANIFEST and SPINE section of the content.opf file. |

# 5.7 — Misc pages

Aim of this plugin is to help the user to make an epub book as complete as possible.

The tab ‘Misc Pages’ allows the user to create the following pages:

* title
* half title
* copyright
* dedication
* also by the same author
* epigraph
* table of content



Picture 5.7-1

The tab has two areas:

1. Input area, where the user enters the values of the page to load into Sigil.
2. Preview area, where the plugin displays the generated code of the page.

**Page operations**.

The user starts the page creation by selecting a page from the ‘choose page type’ combo and then clicking on the ‘add block’ button.

The ‘add block’ button creates a block of empty input fields and displays them in the grid of the ‘input area’. The grid has two columns. The first one (‘row’) serves as user's guide. The user must input the required values in the cells of the second column (‘value’).

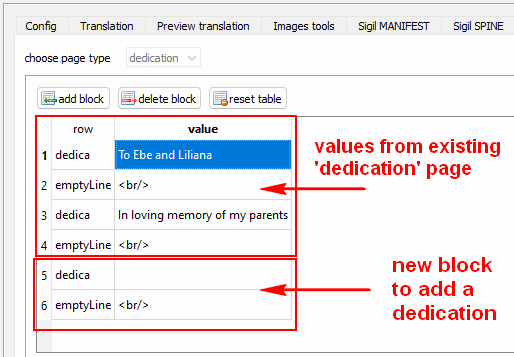
If the chosen page already exists in Sigil, the plugin fills the grid with the values formerly given by the user. If the chosen page does not exists, the plugin displays an empty grid.

Example for the ‘title’ page:

|  |  |
| --- | --- |
| Empty grid (page not existing in Sigil) | Filled grid (page already existing in grid) |
|  |  |

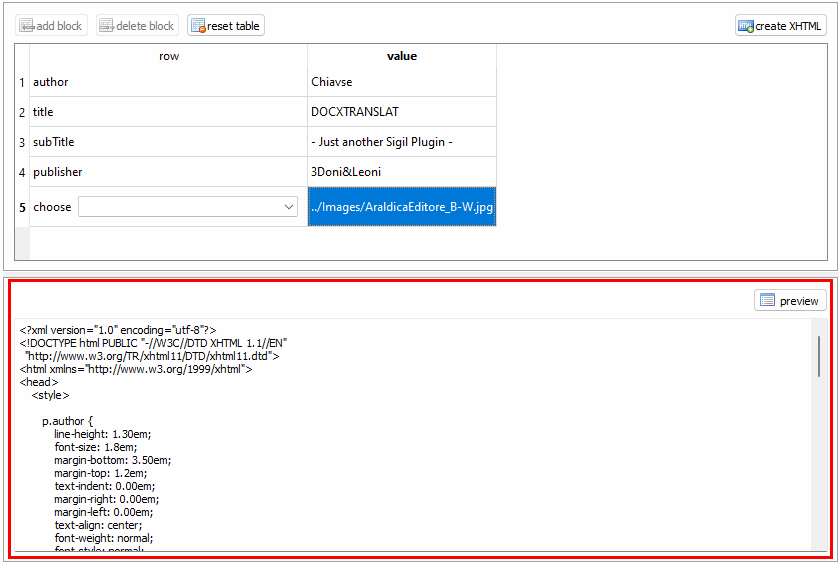
The input fields’ number varies from page to page. Some page only allows one block while others allow many of them (see § 5.7.1 and following).

Example of grid with many input blocks.



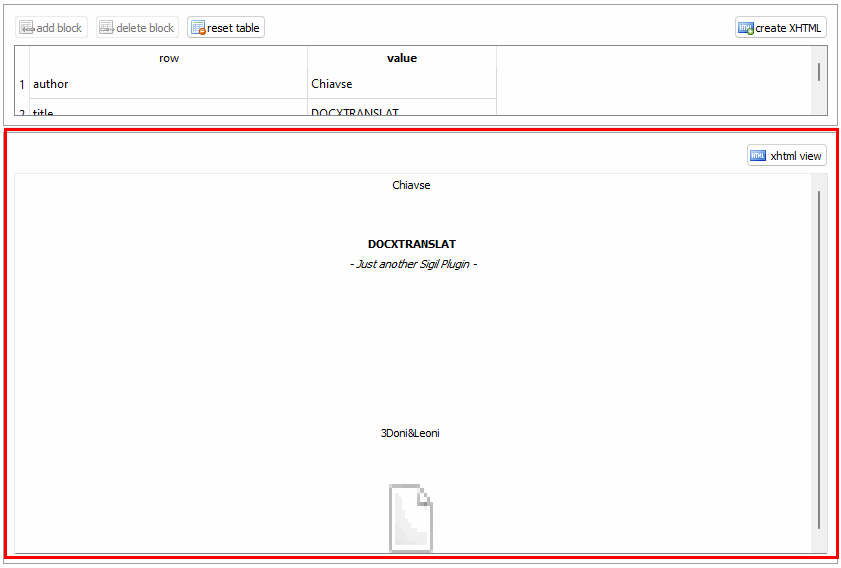
Picture 5.7—2 New input block

After completing the input, the user creates the code of the page by clicking the ‘create XHTML' button. The plugin displays the code of the page in the ‘Preview area’.



Picture 5.7—3 Page preview (code)

By clicking on the ‘preview’ button, the user switches between code view and HTM view of the ‘Preview area’ content.



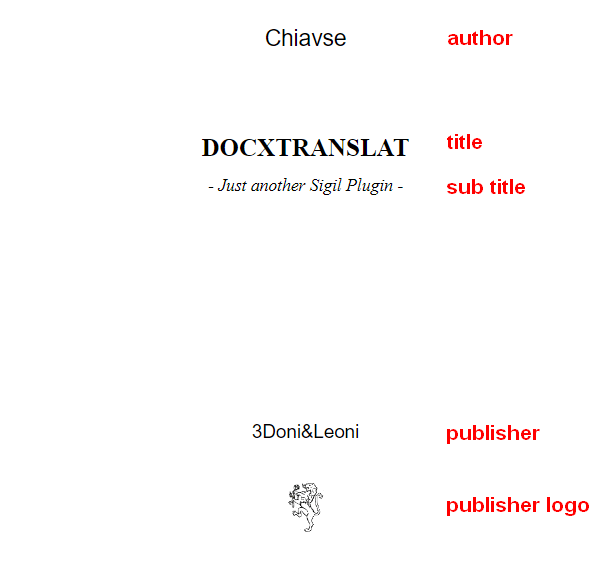
Picture 5.7—4 Page preview (text)

The user can repeat the process of adding/ modifying/ deleting input block and of code creating/ previewing until the page is suited to his requirements. Then the user can load the page into Sigil by clicking the ‘load page into Sigil’ button.

User interface:

|  |  |
| --- | --- |
|  | Chooses the page to create. |
|  | When clicked  1. Disables ‘choose page type’ combo  2. Creates a block of empty input fields and displays them in the grid of the ‘input area’.  For the pages ‘title’, ‘half’, ‘TOC’, only one block allowed. For these pages, after the first click, the ‘add block’ button disables itself and the ‘delete block’ button.  The pages 'copyright', 'dedication', 'also by the same author', 'epigraph', allows the creation of many empty input blocks. For these pages, the ‘add block’ button is always available |
|  | Deletes the input block currently selected in the grid. |
|  | Deletes the grid of the ‘input area’.  Enables ‘add block’ button, ‘delete block’ button and ‘choose page type’ combo. |
|  | Creates the XHTML code of the pages when there is at least one input block (full or empty) in the grid. |
|  | Switches between code view and HTM view of the ‘Preview area’ content |
|  | Load the page into Sigil |

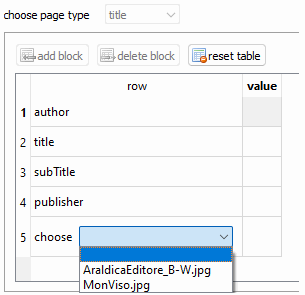
# 5.7.1 — Title page



Picture 4-7-1 Title page example

**Page name in Sigil**: title.xhtml

**Input value grid (empty):**



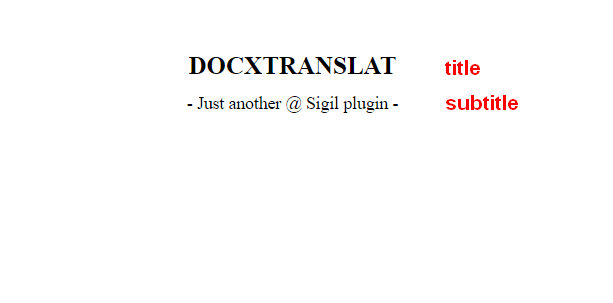
The combo box in the last line of the grid holds the list of all images (if any) already loaded into Sigil. If the user chooses an image from the list, that image will appear in the ‘title’ page as the publisher logo.

**Number of allowed input block**: one

**Operation to create the page:**

1. Select ‘title’ in ‘choose page type’ combo box.
2. Click on ‘add block’ button
3. The input value grid appears in the ‘input area’. ‘add block’ button, ‘delete block’ button and ‘choose page type’ combo are disabled. ‘reset table’ button is enabled.
4. Input values.
5. Click on ‘create XHTML’ button to display the page code in the ‘Display area’
6. Click on ‘load page into Sigil’ to load the code as a new Sigil page. If the page ‘title.xhtml’ is already present in Sigil, the plugin requires confirmation for the operation. After loading, the user can see the name of the page and move it to its correct position using the arrow buttons in the ‘Sigil SPINE’ tab.
7. To end the operations, click on ‘reset table’ button. The grid in the input area disappears. ‘add block’, ‘delete block’ and ‘reset table’ buttons are enabled as well as the ‘choose page type’ combo. The user can execute this operation at each point of the previous sequence.

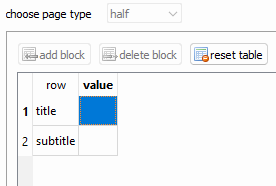
# 5.7.2 — Half title page



Picture 4-7-2 Half title page

**Page name in Sigil:** half.xhtml

**Input value grid (empty):**

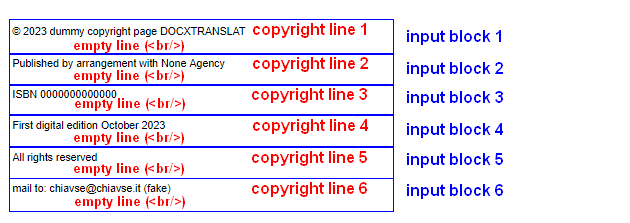
****

**Number of allowed input block**: one

**Operation to create the page:**

1. Select ‘half’ in ‘choose page type’ combo box.
2. Click on ‘add block’ button.
3. The input value grid appears in the ‘input area’. ‘add block’ button, ‘delete block’ button and ‘choose page type’ combo are disabled. ‘reset table’ button is enabled.
4. Input values.
5. Click on ‘create XHTML’ button to display the page code in the ‘Display area’
6. Click on ‘load page into Sigil’ to load the code as a new Sigil page. If the page ‘half.xhtml’ is already present in Sigil, the plugin requires confirmation for the operation. After loading, the user can see the name of the page and move it to its correct position using the arrow buttons in the ‘Sigil SPINE’ tab.
7. To end the operations, click on ‘reset table’ button. The grid in the input area disappears. ‘add block’, ‘delete block’ and ‘reset table’ buttons are enabled as well as the ‘choose page type’ combo. The user can execute this operation at each point of the previous sequence.

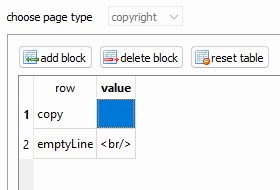
# 5.7.3 — Copyright page



Picture 4-7-3 Copyright page

**Page name in Sigil:** copyright.xhtml

**Input value grid (empty):**



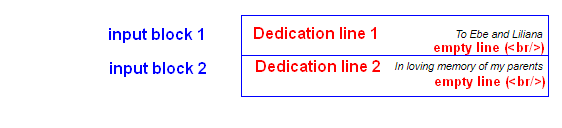
Each input block has two line. First line is for input. Second line is a not modifiable empty line.

**Number of allowed input block**: many

**Operation to create the page:**

1. Select ‘copyright’ in ‘choose page type’ combo box.
2. Click on ‘add block’ button.
3. The input value grid appears in the ‘input area’. ‘add block’, ‘delete block’ and reset table’ are enabled.
4. Input values.
5. Repeat points 2, 3, 4 as many times as there are lines on the copyright page
6. Click on ‘create XHTML’ button to display the page code in the ‘Display area’
7. Click on ‘load page into Sigil’ to load the code as a new Sigil page. If the page ‘copyright.xhtml’ is already present in Sigil, the plugin requires confirmation for the operation. After loading, the user can see the name of the page and move it to its correct position using the arrow buttons in the ‘Sigil SPINE’ tab.
8. To end the operations, click on ‘reset table’ button. The grid in the input area disappears. ‘add block’, ‘delete block’ and ‘reset table’ buttons are enabled as well as the ‘choose page type’ combo. The user can execute this operation at each point of the previous sequence.

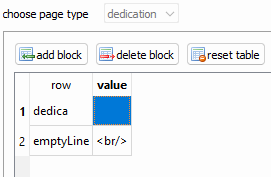
# 5.7.4 — Dedication page



Picture 4-7-4 Dedication page

**Page name in Sigil:** dedication.xhtml

**Input value grid (empty):**



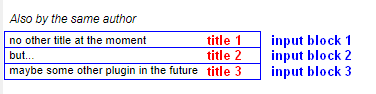
Each input block has two line. First line is for input. Second line is a not modifiable empty line.

**Number of allowed input block**: many

**Operation to create the page:**

1. Select ‘dedication in ‘choose page type’ combo box.
2. Click on ‘add block’ button.
3. The input value grid appears in the ‘input area’. ‘add block’, ‘delete block’ and reset table’ are enabled.
4. Input values.
5. Repeat points 2, 3, 4 as many times as there are lines on the dedication page
6. Click on ‘create XHTML’ button to display the page code in the ‘Display area’
7. Click on ‘load page into Sigil’ to load the code as a new Sigil page. If the page ‘dedication.xhtml’ is already present in Sigil, the plugin requires confirmation for the operation. After loading, the user can see the name of the page and move it to its correct position using the arrow buttons in the ‘Sigil SPINE’ tab.
8. To end the operations, click on ‘reset table’ button. The grid in the input area disappears. ‘add block’, ‘delete block’ and ‘reset table’ buttons are enabled as well as the ‘choose page type’ combo. The user can execute this operation at each point of the previous sequence.

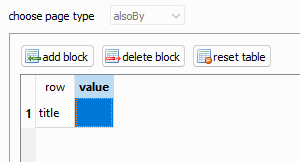
# 5.7.5 — Other titles by the same author



Picture 4-7-5 Also by page

**Page name in Sigil:** alsoBy.xhtml

**Input value grid (empty):**



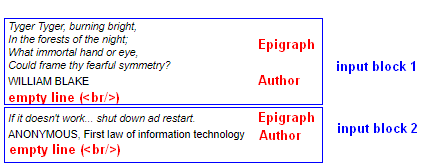
Each input block has one input line.

**Number of allowed input block**: many

**Operation to create the page:**

1. Select ‘alsoBy’ in ‘choose page type’ combo box.
2. Click on ‘add block’ button.
3. The input value grid appears in the ‘input area’. ‘add block’, ‘delete block’ and reset table’ are enabled.
4. Input values.
5. Repeat points 2, 3, 4 as many times as there are lines on the ‘alsoBy’ page
6. Click on ‘create XHTML’ button to display the page code in the ‘Display area’
7. Click on ‘load page into Sigil’ to load the code as a new Sigil page. If the page ‘alsoBy.xhtml’ is already present in Sigil, the plugin requires confirmation for the operation. After loading, the user can see the name of the page and move it to its correct position using the arrow buttons in the ‘Sigil SPINE’ tab.
8. To end the operations, click on ‘reset table’ button. The grid in the input area disappears. ‘add block’, ‘delete block’ and ‘reset table’ buttons are enabled as well as the ‘choose page type’ combo. The user can execute this operation at each point of the previous sequence.

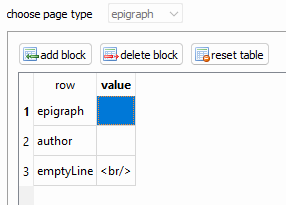
# 5.7.6 — Epigraph page



Picture 4-7-6 Epigraph page

**Page name in Sigil:** epigraph.xhtml

**Input value grid (empty):**

****

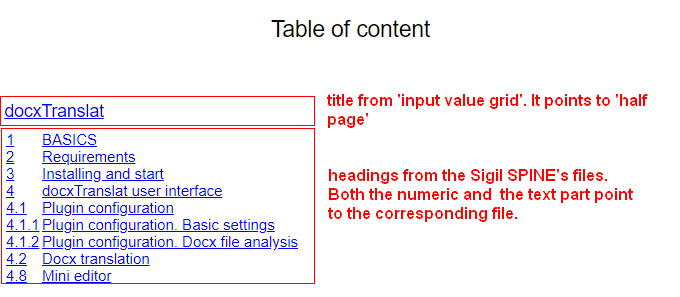
Each input block has three lines. First two lines are for input. The third line is a not modifiable empty line.

**Number of allowed input block**: many

**Operation to create the page:**

1. Select ‘epigraph’ in ‘choose page type’ combo box.
2. Click on ‘add block’ button.
3. The input value grid appears in the ‘input area’. ‘add block’, ‘delete block’ and reset table’ are enabled.
4. Input values.
5. Repeat points 2, 3, 4 as many times as there are epigraphs on the ‘epigraph’ page
6. Click on ‘create XHTML’ button to display the page code in the ‘Display area’
7. Click on ‘load page into Sigil’ to load the code as a new Sigil page. If the page ‘epigraph.xhtml’ is already present in Sigil, the plugin requires confirmation for the operation. After loading, the user can see the name of the page and move it to its correct position using the arrow buttons in the ‘Sigil SPINE’ tab.
8. To end the operations, click on ‘reset table’ button. The grid in the input area disappears. ‘add block’, ‘delete block’ and ‘reset table’ buttons are enabled as well as the ‘choose page type’ combo. The user can execute this operation at each point of the previous sequence.

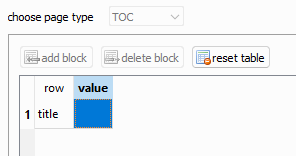
# 5.7.7 — Table of content page (TOC)



Picture 4-7-7 Table of content example

**Page name in Sigil:** customTOC.xhtml

**Input value grid (empty):**

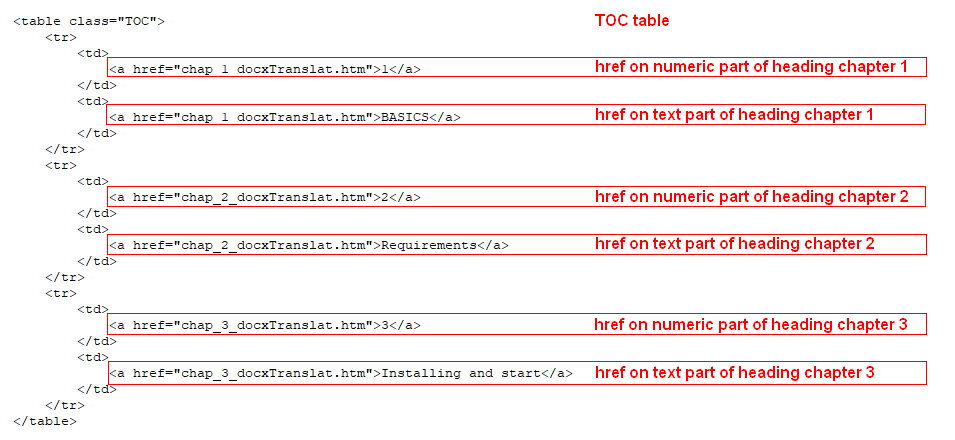


**Number of allowed input block**: one

**Operation to create the page:**

1. Select ‘TOC’ in ‘choose page type’ combo box.
2. Click on ‘add block’ button.
3. The input value grid appears in the ‘input area’. ‘add block’ button , ‘delete block’ button and ‘choose page type’ combo are disabled. ‘reset table’ button is enabled.
4. Input value.
5. Click on the ‘create XHTML’ button to start a process which extracts the heading of each chapter of the document, i.e. of each file contained in the SPINE section of content.opf (§ 5.6). According to the DOCX document requirements (§ 3 — On DOCX files formatting) there is one and only one heading for each file. The search process splits the heading into two parts (numeric and text), inserts both of them into a table and associates each part with a hypertext reference pointing to the source file (see TOC page example below)
6. Click on ‘load page into Sigil’ to load the table of contents as a new Sigil page. If the page ‘customTOC.xhtml’ is already present in Sigil, the plugin requires confirmation for the operation. After loading, the user can see the name of the page and move it to its correct position using the arrow buttons in the ‘Sigil SPINE’ tab.
7. To end the operations, click on ‘reset table’ button. The grid in the input area disappears. ‘add block’, ‘delete block’ and ‘reset table’ buttons are enabled as well as the ‘choose page type’ combo. The user can execute this operation at each point of the previous sequence.

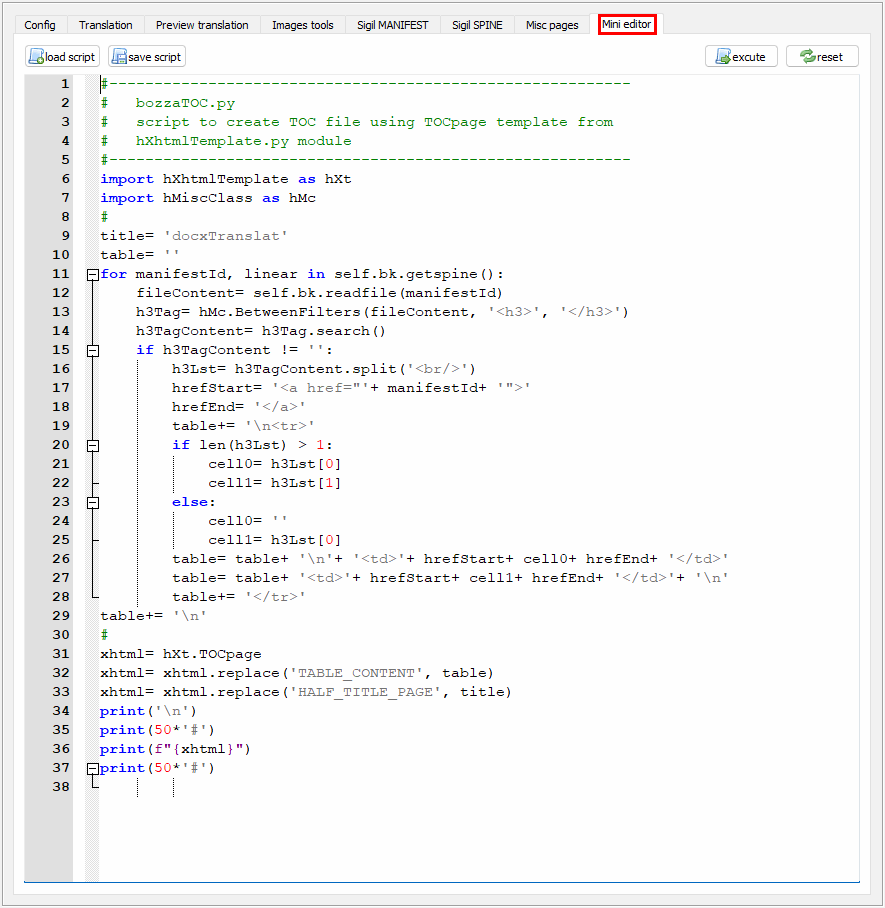
**TOC page code example**



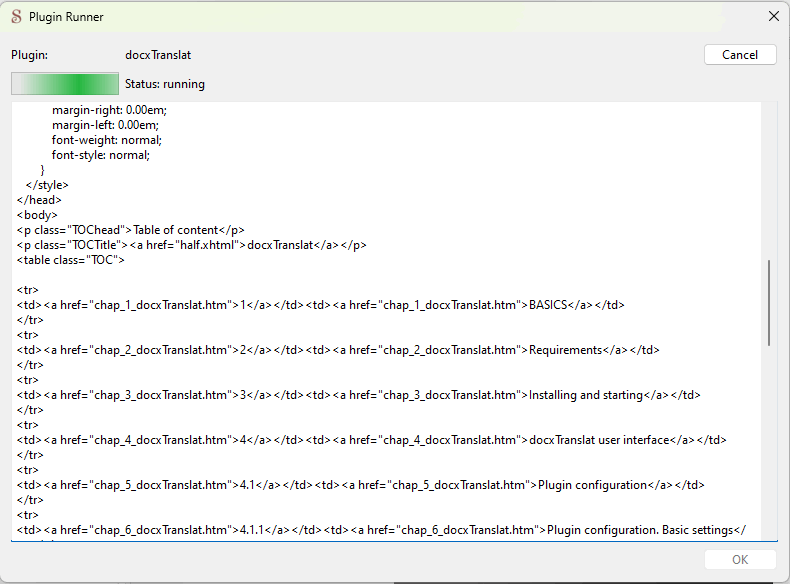
# 5.8 — Mini editor

This tab gives the user access to a simple editor for developing scripts written in the Python programming Language.

The user can write his own code in the ‘Tab area’ of the page, execute it and see the output in the ‘Plugin runner’ page of Sigil.



Picture 5.8—1 Script example



Picture 5.8—2 previous picture script output

The mini editor allows the user to access the Sigil BookContainer class (access to it from the mini editoras **self.bk**). It is also possible to interact with the modules developed to write the docxTranslat plugin (hXhtmlTemplate and hMiscClass in picture 5.8—1).

User interface:

|  |  |
| --- | --- |
|  | Opens a dialog window to allow the user to load a script. |
|  | Opens a dialog window to allow the user to save current script. |
|  | Executes the current script. The user can see the script output in the ‘Plugin Runner’ page. |
|  | Deletes the current script. |

# Acknowledgements

**KevinH,** Sigil Developer. I owe to his manual ‘Sigil\_Plugin\_Framework\_rev12.epub’ all I know about Sigil’s plugins.

<https://www.mobileread.com/forums/showthread.php?t=251452>

Post on 11-20-2014, 03:26 PM

**The Stack Overflow community**, where I have found countless tips on using Python and Pyqt5.

**Martin Fitzpatrick**, for the code I used in hThumbsPreview.py module.

I borrowed Martin’s code from:

<https://www.pythonguis.com/faq/file-image-browser-app-with-thumbnails/>

and from

<https://www.pythonguis.com/faq/remove-and-insertrow-for-martin-fitzpatricks-example>

I made some small changes to Martin's code (mistakes, if any, are my faults).

**Mark James**, for his famfamfam-silk-icons set.

<http://www.famfamfam.com/lab/icons/silk/>

<https://github.com/markjames/famfamfam-silk-icons>

**Detlev Offenbach**, for his Eric IDE: without it the development would have been much more difficult.

<https://eric-ide.python-projects.org/index.html>

