novelWriter Documentation

Release 0.10.2

Veronica Berglyd Olsen

Jul 29, 2020

Contents

This is the documentation for novelWriter 0.10.2.

Contents

CHAPTER 1

Introduction

novelWriter is a simple, multi-document plain text editor using a modified markdown syntax to apply simple formatting. Additional features that are not standard markdown are available through special meta data keywords. These keywords make it possible to inter-link documents, and generate an overview of the entire novel project and how the various files are interconnected.

1.1 Design Philosophy

The user interface is intended to be as minimalistic as practically possible, while at the same time provide a complete set of features needed for writing a novel.

Note: novelWriter is not intended to be a full office type word processor. It doesn't support images, links, tables, and its formatting is limited to headers, and bold, italicised and underlined text.

Most features are accessible through the menu and through keyboard shortcuts. The colour scheme of the user interface can be modified with various themes, and new themes are fairly straight forward to add.

The project itself is laid out in a tree view on the left hand side of the main window. It has various sections, called *root folders*, for the various types of supporting files that the user may want to add to the project. The novel itself lives under its own root folder.

1.2 Project Layout

The layout of the novel itself is managed through the four supported heading levels, H1 through H4. H1 is used for the book title, and for partitions. H2 is used for chapter tiles. H3 is reserved for scene titles. H4 is for section titles within scenes, if such granularity is necessary.

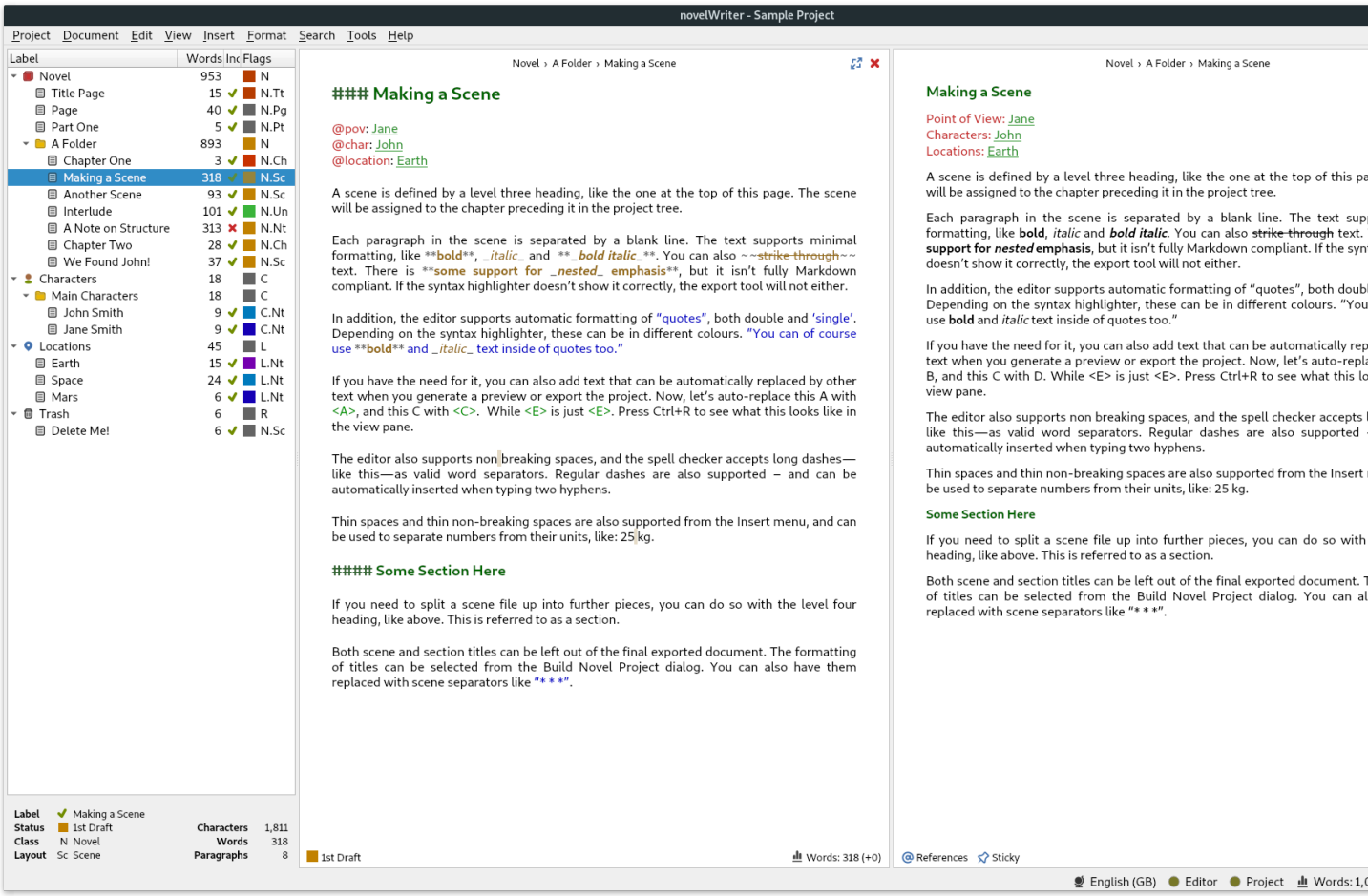
For the files designated as project notes, the usage of headers imply no structural meaning, and the user is free to do whatever they want.

1.3 Project Export

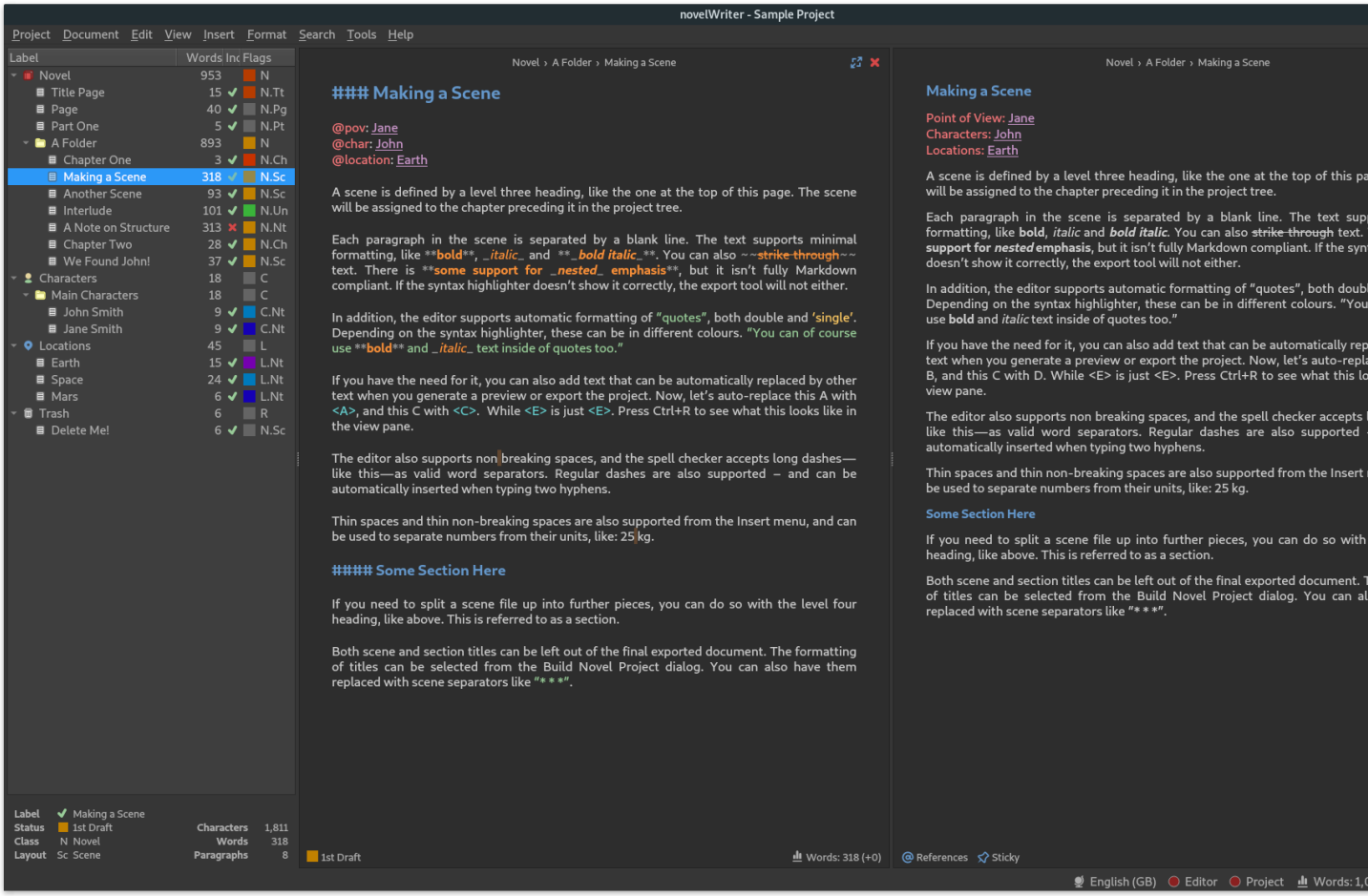
The project can at any time be exported to a range of different formats. Natively, novelWriter supports export to plain text file, HTML document, novelWriter flavoured markdown, standard markdown (requires Qt 5.14), and to a basic Open Document. In addition, printing and printing to PDF is also possible. The best supported export format is HTML, which can be imported or converted by a number of other tools like Pandoc, or simply imported into Libre Office and similar.

1.4 Screenshot

novelWriter with default system theme:



novelWriter with dark theme:


novelWriter - Sample Project
Project Document Edit View Insert Format Search Tools Help
Label Words Inc Flags
Novel 953 N
Title Page 15 N.Tt
Page 40 N.Pg
Part One 5 N.Pt
A Folder 893 N
Chapter One 3 N.Ch
Making a Scene 318 N.Sc
Another Scene 93 N.Sc
Interlude 101 N.Un
A Note on Structure 313 N.Nt
Chapter Two 28 N.Ch
We Found John! 37 N.Sc
Characters 18 C
Main Characters 18 C
John Smith 9 C.Nt
Jane Smith 9 C.Nt
Locations 45 L
Earth 15 L.Nt
Space 24 L.Nt
Mars 6 L.Nt
Trash 6 R
Delete Me! 6 N.Sc
Label Making a Scene
Status 1st Draft
Class N Novel
Layout Sc Scene
Characters 1,811
Words 318
Paragraphs 8
Novel › A Folder › Making a Scene
Making a Scene
@pov: Jane
@char: John
@location: Earth
A scene is defined by a level three heading, like the one at the top of this page. The scene will be assigned to the chapter preceding it in the project tree.
Each paragraph in the scene is separated by a blank line. The text supports minimal formatting, like **bold**, _italic_ and **_bold italic_**. You can also ~~strike through~~ text. There is **some support for _nested_ emphasis**, but it isn't fully Markdown compliant. If the syntax highlighter doesn't show it correctly, the export tool will not either.
In addition, the editor supports automatic formatting of "quotes", both double and 'single'. Depending on the syntax highlighter, these can be in different colours. "You can of course use **bold** and _italic_ text inside of quotes too."
If you have the need for it, you can also add text that can be automatically replaced by other text when you generate a preview or export the project. Now, let's auto-replace this A with <A>, and this C with <C>. While <E> is just <E>. Press Ctrl+R to see what this looks like in the view pane.
The editor also supports non breaking spaces, and the spell checker accepts long dashes—like this—as valid word separators. Regular dashes are also supported – and can be automatically inserted when typing two hyphens.
Thin spaces and thin non-breaking spaces are also supported from the Insert menu, and can be used to separate numbers from their units, like: 25 kg.
Some Section Here
If you need to split a scene file up into further pieces, you can do so with the level four heading, like above. This is referred to as a section.
Both scene and section titles can be left out of the final exported document. The formatting of titles can be selected from the Build Novel Project dialog. You can also have them replaced with scene separators like "* * *".
1st Draft
Words: 318 (+0)
Novel › A Folder › Making a Scene
Making a Scene
Point of View: Jane
Characters: John
Locations: Earth
A scene is defined by a level three heading, like the one at the top of this pa
will be assigned to the chapter preceding it in the project tree.
Each paragraph in the scene is separated by a blank line. The text sup
formatting, like bold, italic and bold italic. You can also strike through text.
support for nested emphasis, but it isn't fully Markdown compliant. If the syn
doesn't show it correctly, the export tool will not either.
In addition, the editor supports automatic formatting of "quotes", both doub
Depending on the syntax highlighter, these can be in different colours. "You
use bold and italic text inside of quotes too."
If you have the need for it, you can also add text that can be automatically rep
text when you generate a preview or export the project. Now, let's auto-repl
B, and this C with D. While <E> is just <E>. Press Ctrl+R to see what this lo
view pane.
The editor also supports non breaking spaces, and the spell checker accepts
like this—as valid word separators. Regular dashes are also supported
automatically inserted when typing two hyphens.
Thin spaces and thin non-breaking spaces are also supported from the Insert
be used to separate numbers from their units, like: 25 kg.
Some Section Here
If you need to split a scene file up into further pieces, you can do so with
heading, like above. This is referred to as a section.
Both scene and section titles can be left out of the final exported document.
of titles can be selected from the Build Novel Project dialog. You can al
replaced with scene separators like "* * *".
References
Sticky
English (GB)
Editor
Project
Words: 1,

CHAPTER 2

Getting Started

You can download novelWriter from https://github.com/vkbo/novelWriter/releases

Latest version is 0.10.2.

2.1 Installing Dependencies

If you already have Python installed, all you need to do is install the dependencies. To do this, your need to open your command line tool, find the folder where you extracted novelWriter, and run:

```
python -m pip install -r requirements.txt
```

On some operating systems you need to use `python3` instead of `python`.

The following Python packages are required to run novelWriter:

- `pyqt5` for the GUI
- `lxml` for writing project files

Note: Sometimes the SVG graphics package for pyqt5 must be installed separately.

The following are optional, but recommended:

- `pyenchant` for spell checking

PyQt/Qt should be at least 5.2.1, but ideally 5.10 or higher for nearly all features to work. Exporting to standard Markdown requires PyQt/Qt 5.14. There are no known minimum for package lxml, but the code was originally written with 4.2. The optional spell check library must be at least 3.0.0 to work with Windows. On Linux, 2.0.0 also works fine.

2.2 Running novelWriter

If all the required dependencies are met, you can run novelWriter from the command line in one of the following ways:

```
python novelWriter.py
python3 novelWriter.py
./novelWriter.py
```

A few switches are supported from the command line, mostly to assist in debugging if an error is encountered. To list all options, run:

```
python novelWriter.py --help
```

There are also a couple of install scripts in the assets folder which will assist in setting up launch icon and the novelWriter project file mimetype for Gnome desktops on Linux. Currently, there's one script for Debian and one for Ubuntu.

2.3 Building a Standalone Executable

A standalone executable can be built with pyinstaller, using the provided python script "install.py" in the source folder. This script will automatically try to install all dependencies and build the standalone executable of novelWriter. You can run the script by typing the following into your command prompt:

```
python install.py
```

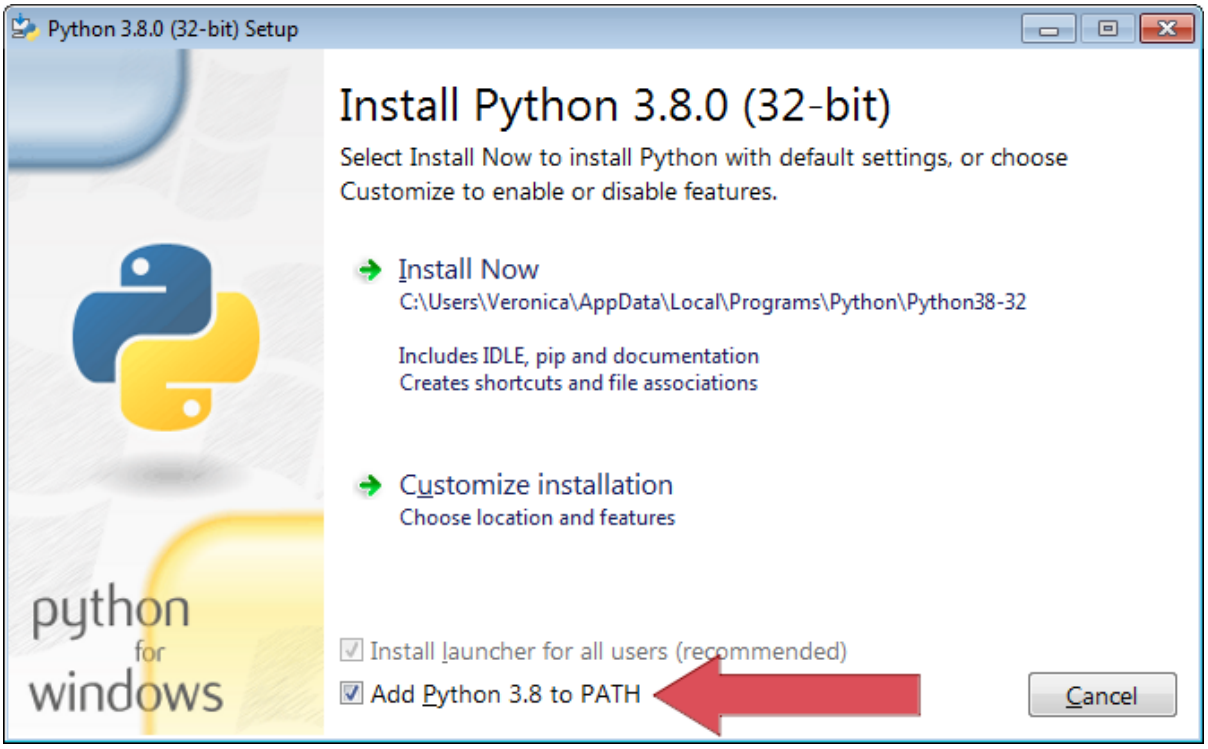
If successful, the executable will be in the "dist" folder.

2.3.1 Additional Instructions for Windows

If you don't have Python installed, you can download it from the python.org website. The installers for Windows are available at https://www.python.org/downloads/windows/

novelWriter should work with Python 3.5 or higher, and the executable installer is the easiest to install. Please note that the *pyenchant* package for spell checking does not currently work with the x86-64 version, so if you want spell checking, you must install the x86 version.

Also, make sure you select the "Add Python to PATH" option.



Once Python is set up and running, you can either run novelWriter from the folder where you extracted it, or you can build an executable and run that from a desktop icon instead.

CHAPTER 3

User Interface

The user interface is kept as simple as possible to avoid distractions when writing. The main window contains a tree view pane with the entire structure of the project, and a small details panel below it to display additional information about the currently selected item.

3.1 Edit View

Editing a document can be done by either double-clicking on it, or hitting the return key when a file is selected. This will open the document editor, which uses a simplified markdown format, described in the section below.

The document currently being edited can also be viewed in parallel in a right hand side view pane. To view a document, simply press `Ctrl-R`, or select a file and go to *Document* → *View Document* in the menu. The document viewed does not need to be the same document currently being edited. If you are viewing the same document as the one you're editing, pressing `Ctrl-R` again will update the document with your last changes.

References to tags can be opened in the view pane from the document editor by moving the cursor to a reference to a tag and hitting `Ctrl-Enter`. In the view panel, the references become clickable links, and the "Referenced By" panel at the bottom will show links to all documents referring back to it.

Note: The "Referenced By" panel relies on an up-to-date index of the project. If anything is missing, or seems wrong, the index can always be rebuilt from *Tools* → *Rebuild Index* or by pressing `F9`.

Both the document editor and the viewer will show the label of the document as set in the Project Tree. Optionally, the full project path to the file can be shown. This can be set the Preferences.

Clicking on the document title bar will select and reveal the file in the Project Tree, making it easier to find the project location of the file in a large project.

3.2 Markdown Format

The document editor uses a simplified markdown format. That is, it supports basic formatting like emphasis (italic), strong emphasis (bold) and strikethrough text, as well as four levels of headings. It is commonly recommended style to differentiate between strong emphasis and emphasis by using `**` for strong emphasis and _ for emphasis, although Markdown generally supports also ___ for strong emphasis and `*` fdr emphasis. However, since the differentiation makes the highlighting and conversion significantly simpler and faster, in novelWriter this is a rule, not just a recommendation.

In addition to these standard markdown features, the editor also allows for comments, that is text that is ignored by the word counter and not exported or, optionally, hidden in the document viewer. If the first word of a comment is "Synopsis:" (with the colon), the comment is treated specially, and will show up in the Outline View. The editor also has a minimal set of keywords used for setting tags and references between files.

Table 1: Formatting Syntax

Format	Description
`# Title`	Heading level one. The space after the # is mandatory.
`## Title`	Heading level two. The space after the # is mandatory.
`### Title`	Heading level three. The space after the # is mandatory.
`#### Title`	Heading level four. The space after the # is mandatory.
`_text_`	The text is rendered as emphasised text (italicised).
`**text**`	The text is rendered as strongly emphasised text (bold).
`~~text~~`	Strikethrough text.
`% text...`	A comment. The text is not exported by default, seen in viewer, or counted towards word counts.
`% Synopsis: text...`	A synopsis comment. Shows up in the Synopsis column of the Outline View, but is otherwise treated as a comment.
`@keyword: value`	A keyword argument followed by a value, or a comma separated list of values.

Some additional rules:

1. The emphasis and strikethrough formatting tags do not allow spaces between the words and the tag itself. That is, `**text**` is valid, `**text **` is not.
2. More generally, the delimiters must be on the outer edge of words. That is, `some **text in bold** here` is valid, `some** text in bold** here` is not.
3. If using both `**` and _ to wrap the same text, the underscore must be the inner wrapper. This is due to the underscore also being a valid word character, so if they are on the outside, they violate rule 2.

The editor and viewer also supports markdown standard hard line breaks, and preserves non-breaking spaces. A hard line break is achieved by leaving two or more spaces at the end of the line. Alternatively, the user can press `Ctrl-K, Return` to insert this. A non-breaking space is inserted with `Ctrl-K, Space`.

Thin spaces are also supported, and can be inserted with `Ctrl-K, Shift-Space`, and the non-breaking version of it with `Ctrl-K, Ctrl-Space`.

Both hard line breaks and non-breaking spaces are highlighted by the syntax highlighter as an alternate coloured background, depending on the selected theme.

3.3 Project Outline View

The Project Outline View is available as the second tab on the right hand side of the main window marked "Outline". The Outline View provides an overview of the novel structure, displaying a tree hierarchy of the elements of the novel, that is, the level 1 to 4 headings.

Various meta data and information extracted from tags can be displayed in columns in the Outline View. To turn on or off specific columns, right click the header and select the columns you want to show. The order of the columns can be rearranged by dragging them to a different position.

Note: The "Title" columns cannot be disabled or moved.

The information viewed in teh Outline View is based on the Project Index. While novelWriter does its best to keep the index up-to-date when content changes, you can always rebuild it manually by pressing `F9`.

The Outline View itself can be regenerated by pressing `F10`. You can also enable automatic updating in the *Tools* menu, which will trigger an update whenever the index is updated. You may want to disable this feature if your project is very large,

3.4 Synopsis Feature

The "Synopsis" column of the Outline View takes its information from a specially formatted comment. In order to flag a comment as a Synopsis, add the word "Synopsis:" as the first word of the comment. The ":" is required, and "synopsis" is not case sensitive. If it is correctly formatted, the syntax highlighter will indicate this by altering the colour of the word.

Note: Only one comment can be flagged as a synopsis comment for each heading. If multiple comments are flagged as a synopsis, the last one will be used.

3.5 Keyboard Shortcuts

Most features are available as keyboard shortcuts. These are as following:

Table 2: Keyboard Shortcuts

Shortcut	Description
`Alt-1`	Switch focus to tree view pane.
`Alt-2`	Switch focus to document editor pane.
`Alt-3`	Switch focus to document viewer pane.
`Ctrl-.`	Correct word under cursor.
`Ctrl-,`	Open the Preferences dialog.
`Ctrl-/`	Change block format to comment.
`Ctrl--`	Strikethrough selected text, or word under cursor.
`Ctrl-0`	Remove block formatting for block under cursor.
`Ctrl-1`	Change block format to header level 1.
`Ctrl-2`	Change block format to header level 2.
`Ctrl-3`	Change block format to header level 3.
`Ctrl-4`	Change block format to header level 4.

Continued on next page

Table 2 – continued from previous page

Shortcut	Description
Ctrl-A	Select all text in document.
Ctrl-B	Format selected text, or word under cursor, with strong emphasis (bold).
Ctrl-C	Copy selected text to clipboard.
Ctrl-D	Wrap selected text, or word under cursor, in double quotes.
Ctrl-E	If in tree view, edit a document or folder settings. (Same as F2)
Ctrl-F	Open the search bar and search for selected word, if any is selected.
Ctrl-G	Find next occurrence of word in current document. (Same as F3)
Ctrl-H	Open the search and replace bar and search for selected word, if any is selected. (On Mac, this is Cmd-=)
Ctrl-I	Format selected text, or word under cursor, with emphasis (italic).
Ctrl-N	Create new document.
Ctrl-O	Open selected document.
Ctrl-Q	Exit novelWriter.
Ctrl-R	If in tree view, open a document for viewing. If editor pane has focus, open current document for viewing.
Ctrl-S	Save the current document in the editor.
Ctrl-V	Paste text from clipboard to cursor position.
Ctrl-W	Close the current document in the editor.
Ctrl-X	Cut selected text to clipboard.
Ctrl-Y	Redo latest undo.
Ctrl-Z	Undo latest changes.
Ctrl-F7	Toggle spell checking.
Ctrl-F10	Toggle automatic updating of project outline.
Ctrl-Del	If in tree view, move a document to trash, or delete a folder.
Ctrl-Enter	Open the tag or reference under the cursor in the view panel.
Ctrl-Shift-,	Open the Project Settings dialog.
Ctrl-Shift-/	Remove block formatting for block under cursor.
Ctrl-Shift-1	Replace occurrence of word in current document, and search for next occurrence.
Ctrl-Shift-A	Select all text in current paragraph.
Ctrl-Shift-B	Format selected text, or word under cursor, with very strong emphasis (bold and italic).
Ctrl-Shift-D	Wrap selected text, or word under cursor, in single quotes.
Ctrl-Shift-G	Find previous occurrence of word in current document. (Same as Shift-F3
Ctrl-Shift-I	Import text to the current document from a text file.
Ctrl-Shift-N	Create new folder.
Ctrl-Shift-O	Open a project.
Ctrl-Shift-R	Close the document view pane.
Ctrl-Shift-S	Save the current project.
Ctrl-Shift-W	Close the current project.
Ctrl-Shift-Up	Move item one step up in the tree view.
Ctrl-Shift-Down	Move item one step down in the tree view.
F1	Open documentation. This just tries to send the documentation URL ti your browser.
F2	If in tree view, edit a document or folder settings. (Same as Ctrl-E)
F3	Find next occurrence of word in current document. (Same as Ctrl-G)
F5	Open the Build Novel Project dialog.
F6	Open the Writing Statistics dialog.
F7	Re-run spell checker.
F8	Activate Focus Mode, hiding project tree and view panel.
F9	Re-build project index.
F10	Re-build project outline.

Continued on next page

Table 2 – continued from previous page

Shortcut	Description
`F11`	Activate full screen mode.
`Shift-F3`	Find previous occurrence of word in current document. (Same as `Ctrl-Shift-G`
`Enter`	If in tree view, open a document for editing.

Note: On macOS, replace `Ctrl` with `Cmd`.

A set of insert features are also available through shortcuts, but they require a double combination of shortcuts. The insert feature is activated with `Ctrl-K`, followed by a key for the inserted character or punctuation.

Table 3: Keyboard Shortcuts

Shortcut	Description
`Ctrl-K, -`	Insert a short dash (en dash).
`Ctrl-K, _`	Insert a long dash (em dash).
`Ctrl-K, .`	Insert ellipsis.
`Ctrl-K, 1`	Insert left single quote.
`Ctrl-K, 2`	Insert right single quote.
`Ctrl-K, 3`	Insert left double quote.
`Ctrl-K, 4`	Insert right double quote.
`Ctrl-K, Return`	Insert a hard line break.
`Ctrl-K, Space`	Insert a non-breaking space.
`Ctrl-K, Shift-Space`	Insert a thin space.
`Ctrl-K, Ctrl-Space`	Insert a thin non-breaking space.

CHAPTER 4

Novel Projects

A novelWriter project requires a dedicated folder for storing its files on the local file system. See the Technical Information section for further details.

A new project can be created from the Project menu by selecting *Project → New Project*. A list of recently opened projects is maintained, and displayed in the "Open Project" dialog. A project can be removed from this list by selecting it and pressing the `Del` key.

The project specific settings are available in *Project → Project Settings*. See further details below.

4.1 Project Roots

Projects are structured into a set of root folders, visible in the left side tree view panel.

The core novel files go into a root folder of type "Novel". Other supporting files go into root folders of types "Plot", "Characters", "Locations", "Timeline", "Objects", "Entities", or "Custom". These other root folder types are intended for your notes on the various elements of your story. Using these is of course entirely optional. A new project will not have all of the root folders present, but you can add the ones you want from *Project → Create Root Folder*.

The root folders are intended for the following use, but aside from the Novel folder, no restrictions are enforced by the application. You can use them however you want.

Note: The root folders correspond to the categories of tags that can be used. See the "Project Structure" section for further details.

- **Novel:** The root folder of all text that goes into the final novel. This class of files have other rules and features than other files in the project. See the Novel Structure section for more details.
- **Plot:** This is the root folder where main plots can be outlined. It is optional, but adding at least dummy files can be useful in order to tag plot elements for the Outline View.
- **Characters:** Character files go in this root folder. These are especially important if one wants to use the Outline View to see which character appears where, and which part of the story is told from a specific character's point-of-view.

- **Locations:** Location is for various scene locations that one wants to track.
- **Timeline:** If the story jumps in time within the same plot, this class of files can be used to track this.
- **Objects:** Important objects in the story can be tracked here.
- **Entities:** Entities, like organisations or companies, that are part of the plot, can be organised here.
- **Custom:** The custom root folder can be used for tracking anything else not covered by the above options.

Deleted files will be moved into a special "Trash" root folder. Files in the Trash folder can be deleted permanently.

4.1.1 Orphaned Documents

In the event the editor crashes or otherwise exits without saving the project state, files that have been added to the project tree and are saved to disk will appear in a special "Orphaned Items" root folder next time the application is started. These orphaned files will not have any meta data associated with them, although novelWriter will try to restore the file label it had in the project tree. Other information will have to be set again, and the files moved back to the correct location in the project.

4.1.2 Project Lockfile

To prevent orphaned files caused by file conflicts when novelWriter projects are synced with file synchronisation tools, a project lockfile is written to the project folder. If you try to open a project which has such a file, you will be presented with a warning, and some information about where novelWriter thinks the project is open. You will be give the option to ignore this warning, and continue opening the project. However, if multiple instances are in fact editing the same project, you are likely to cause inconsistencies and create diverging project files, potentially resulting in loss of data.

Note: If, for some reason, novelWriter crashes, the lock file may remain. In such a case it is safe to ignore the lock file warning when re-opening the project.

4.1.3 Using Folders in the Project Tree

Folders, aside from root folders, have no structural significance to the project. They are there purely as a way for the user to organise the files in meaningful sections and to be able to close them in the tree view. When processing the files in the novel, like for instance during export, the folders are ignored.

4.2 Project Settings

The project settings can be accessed from the *Project* → *Project Settings* menu entry. This will open a dialog box, with a set of tabs.

4.2.1 Settings Tab

The Settings tab holds the project title and author settings. Working Title can be set to a different title than the Book Title. The difference between them is simply that the Working Title is used for the GUI (main window title) and for generating the backup files. The intention is that the working title should remain unchanged, while changing the final title has no effect on features relying on the project name. The Book Title is currently not ues for anything, so setting it is just for the benefit of the author.

The Book Authors text box takes one author per line.

4.2.2 Details Tab

This tab presents an overview of meta data about the project. It states where on your file system the project is saved, how may times it has been saved, how many folders and files it contains, and how many words exist in the entire project.

4.2.3 Status Tab

Each file of type "Novel" can be given a status level, signified by a coloured icon. These are purely there for the user's convenience, and you are not required to use them for any other feature to work. The intention is to use this list to set what stage of writing you are on, although you can in principle make them whatever you want.

Note: The status levels currently in use by a file cannot be deleted.

4.2.4 Importance Tab

Each file of types "Plot", "Character", "World", "Timeline", "Object", "Entity", or "Custom", can be given an importance level, signified by a coloured icon like for status level. These are also purely there for the user's convenience, and you are not required to use them for any other feature to work. The intention is to use this list to set how important the character, plot element, or otherwise, is for the story. Again, these can in principle be used for whatever you want.

Note: The importance levels currently in use by a file cannot be deleted.

4.2.5 Auto-Replace Tab

A set of automatically replaced keywords can be added in this tab. The keywords in the left column wile be replaced by the text in the right column when documents are opened in the viewer. This will also be applied to exports when the feature is added.

Note that a keyword cannot contain any spaces. The angle brackets are added by default, and when used in the text are a part of the keyword to be replaced. This is to ensure that parts of the text isn't unintentionally replaced by the content of the list.

4.3 Writing Files

New document files can be created from the Document menu, or by pressing `Ctrl-N` while in the tree view pane. This will create a new, empty file, and open the item settings dialog where the filename and various other settings can be set. This dialog can also be opened again later from either the menu, *Project* -> *Edit* item, or by pressing `Ctrl-E` or `F2` with the item selected.

The layout of the file is also defined here. For Novel files, the full list of layout options are available. For non-Novel files, only "Note" is available. You can also select whether the file is by default included when building the project. This setting can be overridden in the export tool if you wish to include them anyway.

See the Project Structure section for more details.

4.4 Backup

An automatic backup system is built into novelWriter. In order to use it, a backup path to where the backups are to be stored must to be provided in *Tools* → *Preferences*. Backups can be run automatically when a project is closed, which also implies it is run when the application is closed. Backups are date stamped zip files of the entire project folder, and are stored in a subfolder of the backup path with the same name as the project working title set in Project Settings.

The backup feature, when configured, can also be run manually from the *Tools* menu. It is also possible to dissable automated backup for a given project in Project Settings.

Note: For the backup to be able to run, the Working Title must be set in Project Settings. This value is used to generate the folder name for the zip files.

CHAPTER 5

Project Structure

This section covers the structure of a novel project.

Note: This section concerns files under the Novel type root folder only. There are some restrictions and features that only applies to these type of files.

5.1 Importance of Headings

Subfolders under root folders have no impact on the structure of the novel itself. The structure is instead dictated by the heading level. Four levels of headings are supported, signified by the number of hashes preceding the title. See the Markdown section.

The header levels are not only important when generating the exported novel file, but they are also used by the indexer and Outline View. Each heading starts a new region where new references to tags can be set.

The different header levels are interpreted as specific section types of the novel.

- `# Header1`: Header level 1 signifies that the text refers to either the novel title or the name of a top level partition.
- `## Header2`: Header level 2 signifies a chapter level partition.
- `### Header3`: Header level 3 signifies a scene level partition.
- `#### Header4`: Header level 4 signifies a sub-scene level partition (section).

5.2 Tag References

Each partition, indicated by a heading, can contain references to tags set in the supporting files of the project.

The references are gathered by the indexer and used to generate the Outline View of how the different parts of the novel are connected. References and tags are also clickable in the view panel, and makes it easy to navigate reference

notes while writing. The targets of references can also be set per header. This is covered in the "Supporting Files" section.

References are set as keyword and a list of corresponding tags. The valid keywords are listed below. The format of a meta line is `@keyword: value1, [value2] ... [valueN]`. All keywords allow multiple values.

- `@pov`: The point-of-view character for the current section. The target must be a note tag in the character root folder.
- `@char`: Other characters in the current section. The target must be a note tag in the character root folder. This should not include the point-of-view character.
- `@plot`: The plot timelines touched by the current section. The target must be a note tag in the plot root folder.
- `@time`: The timelines touched by the current section. The target must be a note tag in the timeline root folder.
- `@location`: The location the current section takes place in. The target must be a note tag in the locations root folder.
- `@object`: Objects present in the current section. The target must be a note tag in the object root folder.
- `@entity`: Entities present in the current section. The target must be a note tag in the entities root folder.
- `@custom`: Custom references in the current section. The target must be a note tag in the custom root folder.

The syntax highlighter will alert the user that only the correct keywords are used, and that the tags referenced exist. If the index of defined tags is out of date, press `F9` to regenerate it, or select *Tools* → *Rebuild Index* from the menu. In general, the index for a file is regenerated when a file is saved, so this shouldn't normally be necessary.

5.3 Novel File Layout

Files in a novelWriter project can have a layout format set. These layouts are important when the project is exported, as they indicate how to treat the content in terms of formatting, headings and page breaks. The layout for each file is indicated as the last set of characters in the Flags column of the project tree. They also help to indicate what each file is for in your project.

Some of these layout types are different, some are just cosmetic. The "Book" layout is a generic novel file layout that in formatting is identical to "Chapter" and "Scene", but may help to indicate what files do in your project. You can lay out your project using Book files for each act, and then later split those into chapter or scene files by using the "Split Document" tool. Scenes can also be contained within chapter files, but you lose the drag and drop feature that comes with having them in separate files.

Some layouts have implications on how the project is exported. Files with layout "Title" and "Partition" have all headings and text centred, while the "Unnumbered" layout disables the automatic chapter numbering feature for everything contained within it.

All of the above layout formats are only usable in the Novel root folder. Files that are not a part of the novel itself should have the Note layout. These files are not getting any special formatting, and it is possible to collectively filter them out during export. Note files can be used anywhere in the project.

Below is an overview of all available layout formats.

- **Title Page**: The title page layout. The title should be formatted as a heading level one. All text is automatically centred on exports.
- **Plain Page**: A plain page layout useful for instance for front matter pages. Heading levels are ignored for this layout format, and so are formatting options like Justify Text. The page is exported with a page break before it.
- **Book**: This is the generic novel file format that in principle can be used for all novel files. Since the internal structure of the novel is controlled by the heading levels, this file will produce the same result as a collection

of Partition, Chapter and Scene type files. However, it does not provide the functionality of the Unnumbered layout format.

- **Partition**: A partition can be used to split the novel into parts. Partition titles are indicated with a level one heading. You can also add text and meta data to the page. The Partition file layout will in addition force a page break before the heading, and centre all content on the page.
- **Chapter**: Signifies the start of a new chapter. If the text itself is contained in scene files, these files should only contain the title, comments, synopsis, and tag references for characters, plot, etc. The heading for chapters should be level two. If you need an opening text, like a quote or other leading text before the first scene, this is also where you'd want to add this text.
- **Unnumbered**: Same as Chapter, but when exporting the files and automatic chapter numbering is enabled, this file will not receive a number. This makes the layout suitable for Prologue and Epilogue type chapters.
- **Scene**: A scene file. This file should have a header of level three. Further sections can have headers of level four, but there are no file layout specifically for sections.
- **Note**: A generic file that is optionally ignored when the novel is exported. Use these files for descriptions of content in the supporting root folders. Note files can also be added to the Novel root folder if you need to insert notes there. Note file headers receive no formatting when building the project. They are always exported as-is.

Note: The layout granularity is entirely optional. In principle, you can write the entire novel in a single file with layout "Book". You can also have a single file per chapter.

CHAPTER 6

Supporting Files (Notes)

Supporting files, or notes, are any files stored in root folders that are not the Novel root folder. These files are intended for summaries and outlines of the various plot elements, characters, locations, and so on, of the novel. These are not required, but making at least minimal files for each such element, and add a tag to them, makes it possible to use the Outline View feature to see how each element intersects with each section of the novel itself, and add clickable cross-references between document in the editor and viewer.

6.1 File Tags

Each new heading in a note file can have a tag associated with it. The format of a tag is `@tag: tagname`, where tagname is a unique identifier. Tags can then be referenced in the novel files, or other note files, and will show up in the Outline View and in the back-reference panel when a document is being viewed.

The syntax highlighter will alert the user that the keyword is correctly used and that the tag is allowed, that is, the tag is unique. Duplicate tags should be detected as long as the index is up to date.

The tag is the only part of these files that the application uses. The rest of the file is there for the writer to use in whatever way they wish.

A note file can also reference other note files in the same way novel files do. When the note file is opened in the view pane, these become clickable links, making it easier to follow connections in the plot.

CHAPTER 7

Exporting Projects

The novelWriter project can be exported in various formats using the build tool available from *Project* → *Build Project* or by pressing `F5`.

7.1 Header Formatting

The titles for the four levels of story structure can be formatted collectively in the export tool. This is done through a series of keyword–replace steps.

The keyword `%title%` will always be replaced by the text you put after the # characters in your document.

The keywords `%ch%` and `%chw%` is replaced by a number, or a number word, respectively. You can also use `%chi%` or `%chI%` for lower and upper case Roman numbers. The number is incremented by one each time the build tool sees a new heading of level two in a file with layout "Chapter". If the file has layout "Unnumbered", the counter is *not* incremented. The latter is useful for for instance Prologue and Epilogue chapters.

Likewise, the keywords `%sc%` and `%sca%` are number counters for scene files. These are incremented each time a heading of level three is encountered. The former keyword is reset to one for each new chapter, while the latter is not reset but counts from first scene encountered in the project.

If you want to insert a line break in your title format, add two backslashes `\\`.

Note: Header formatting only applies to novel files. Headings in note files will will be left as-is, but heading levels 1 through 4 are converted to the correct heading level in the respective output formats.

7.2 Scene Separators

If you don't want any titles for your scenes (and for your sections if you have them), you can leave the boxes empty, and an empty paragraph will be inserted between the scenes or sections instead. Alternatively, if you want a separator

between them, like the common "***", you can also enter that in the box. In fact, if the format is a piece of static text, it will always be treated as a separator.

7.3 File Selection

Which files are selected for export can be controlled from the options on the left side of the dialog window. The switch for "Include novel files" will select any file that isn't classified as a note. That is, files with layout "Book", "Page", "Partition", "Chapet", "Unnumbered", or "Scene". The switch for "Include note files" will select any file that is a note. That is, files with layout "Note". This is allows for exporting just the novel, just your notes, or both, as you see fit.

In addition, you can select to export the synopsis comments, regular comments, keywords, and even exclude the body text itself. If you for instance want to export a document with an outline of the novel, you can enable keywords and synopsis export and disable body text, thus getting a document with each heading followed by the tags and references and the synopsis.

If you need to exclude specific files from your exports, like draft files or files you want to take out of your build, but don't want to delete, you can uncheck the "Include when building project" option for each file in the project tree. An included file has a checkmark after the status icon in the "Flags" column. The "Build Novel Project" tool has a switch to ignore this flag if you need to collectively override these settings.

7.4 Export Formats

Currently, six formats are supported for exporting.

7.4.1 OpenDocument Format

This is produces an open document `.odt` file. The document produced has very little formatting, and may require further editing afterwards. For a better formatted office document, you may get a better result with exporting to HTML and the import that HTML document in your office word processor.

7.4.2 PDF Format

The PDF export is just a shortcut for print to file.

7.4.3 novelWriter HTML

The HTML export format writes a single `.htm` file with minimal style formatting. The exported HTML file is suitable for further processing by document conversion tools like Pandoc, for importing in word processors, or for printing from browser.

7.4.4 novelWriter Markdown

This is simply a concatenation of the files selected by the filters. The files in the project are stacked together in the order they appear in the tree view, with comments, tags, etc. included if they are selected. This is a useful format for exporting the project for later import back into novelWriter.

7.4.5 Standard Markdown

If you have Qt 5.14 or higher, the option to export to plain Markdown is available. This feature uses Qt's own Markdown export feature.

7.4.6 Plain Text

The plain text export format writes a simple `.txt` file without any formatting at all.

7.5 Additional Export Options

In addition to the above document formats, the novelWriter HTML and Markdown formats can also be wrapped in a JSON file. The files will have a meta data entry and a body entry. For HTML, also the accompanying css styles are exported.

The text body is saved in a two-level list. The outer list contains one entry per exported file, in the order they appear in the project tree. Each file is then split up into a lst as well, with one entry per line.

These files are mainly intended for scripted post-processing for those who want that option. A JSON file can be imported directly into a Python dict object or a PHP array, to mentions a few options.

CHAPTER 8

Technical Information

This section contains details of how novelWriter stores and handles the project data.

8.1 How Data is Stored

All novelWriter files are written with utf-8 encoding. Since Python automatically converts Unix line endings to Windows line endings on Windows systems, novelWriter does not make any adaptations to the formatting on Windows systems. This is handled entirely by the Python standard library.

8.1.1 Main Project File

The project itself requires a dedicated folder for storing its files, where novelWriter will create its own "file system" where the folder and file hierarchy is described in a project XML file. This is the main project file in the project's root folder with the name `nwProject.nwx`. This file also contains all the meta data required for the project, and a number of related project settings.

If this file is lost or corrupted, the structure of the project is lost. It is important to keep this file backed up, either through the built-in backup tool, or your own backup solution.

Note: The novelWriter project folder is structured so that it can easily be added to a version control system like git. If so, you may want to add a *.gitignore* file to exclude files with the extensions *.json* as JSON files are used to cache the index and various run-time settings.

The project XML file is indent-formatted, suitable for diff tools and version control, although a timesetamp is set in the meta section on line 2 each time the file is saved.

8.1.2 Project Documents

The project documents are saved in a folder in the main project folder named `content`. Each document has a file handle taken from the first 13 characters of a SHA256 hash of the system time when the file was first created. The documents are saved with a filename assembled from this hash and the file extension `.nwd`. If you wish to find the physical location of a file in the project, you can either look it up in the project XML file, or select *Document → Show File Details* in the menu when having the document open.

The reason for this cryptic file naming is to avoid issues with file naming conventions and restrictions on different operating systems, and also to have a file name that does not depend on what the user names the files, or changes it to. The file meta data in the tree view, except the file label, is only saved in the project XML file.

Each document file contains a plain text version of the text from the editor. The file can in principle be edited in any text editor, and is suitable for diffing and version control if so desired. Just make sure the file remains in utf-8 encoding, otherwise unicode chatracters may become mangled when opened in novelWriter again.

The first line of the file contains some meta data starting with the characters "%%~". This line is mainly there to restore some information if it is lost from the project file, and the information may be helpful if you do open the file in an external editor as it contains the file label as the last entry. The line can be deleted without any consequences to the rest of the content of the file, and will be added back next time the file is saved in novelWriter.

8.1.3 The File Saving Process

When saving the project file, or any of the documents, the data is first saved to a temporary file. If successful, the old data file is removed, and the temporary file becomes the new file. This ensures that the previously saved data is only replaced when the new data has been successfully saved. For the project XML file, a *.bak* file is kept which will always contain the previous version of the file, although when auto-save is enabled, they may have the same content.

Indices and Tables

- genindex
- modindex
- search