
SIGIL

Sigil Manual

Release 0.2.0

Strahinja Marković

May 08, 2010

CONTENTS

1	Introduction	1
1.1	Installation	1
2	The main user interface	3
2.1	What are all these files?	3
2.2	The Views	5
3	Find & Replace	9
3.1	Wildcard mode	9
3.2	Regular expression mode	11
4	Book Browser	13
4.1	Content folders	13
4.2	Context menu	15
5	Meta Editor	17
5.1	Collapsed UI	17
5.2	Expanded UI	18
6	Table Of Contents Editor	21
6.1	Building the TOC	21
6.2	The editor interface	23
6.3	Advanced uses	23
7	EPUB Overview	25
7.1	Open Publication Structure	25
7.2	Open Packaging Format	26
7.3	OEBPS Container Format	29
8	Glossary	31

INTRODUCTION

Sigil is a free and open source editor for the EPUB format. It is designed for easy, *WYSIWYG* editing of EPUB files and for converting other formats to EPUB. It also provides features for advanced users, like direct *XHTML*, *CSS* and *XPGT* editing. You can use it to add any of the metadata entries supported by the EPUB specification and create a hierarchical Table of Contents.

The editor works on all three major platforms: Windows, Linux and Mac OS X.

Sigil is a work in progress, and as such is still missing many features. If you have the programming skills, you can directly contribute to its development. Even if you don't, just using Sigil and reporting any bugs or missing features on the [issue tracker](#) will go a long way towards making it a better application for everyone.

1.1 Installation

Installing Sigil is fairly easy on all platforms. The files can be downloaded from the [download section](#) of the project site.

1.1.1 Windows

On Windows, you merely run the the appropriate installer. If you have a 64 bit version of Windows, you should use the Windows installer labeled with `x86_64`. Otherwise, use the vanilla one.

Then just run it. Everything should take care of itself. If you get an "application configuration is incorrect" message when starting Sigil, see [this item](#) in the FAQ.

Sigil is tested on XP, Vista x64 and Windows 7 x64.

1.1.2 Linux

Run the installer for your CPU architecture. Two are available: `x86` and `x86_64`. If in doubt, use the `x86` one. The [Qt Framework](#) is included in the installer.

You have to do this from the console. First, make the installer executable and then run it:

```
chmod +x installer.bin
sudo ./installer.bin
```

After the installer completes, there should be an icon on your desktop and an entry in your “start menu”.

Sigil can be uninstalled by running the `uninstall` executable located in the folder in which Sigil was installed. By default, this is `/opt/sigil`. So with default options, uninstalling Sigil would look like this:

```
cd /opt/sigil
sudo ./uninstall
```

Sigil is tested on Ubuntu 9.10 (Karmic Koala).

1.1.3 Mac OS X

Unpack the DMG file and drag the `Sigil.app` file to your Applications folder.

Sigil is built as a universal binary, and is tested on Mac OS X 10.5 and 10.6. Sigil should also work on Tiger (10.4), but is untested on that version of Mac OS X.

THE MAIN USER INTERFACE

Sigil provides an advanced editing environment for EPUB books. The main user interface (UI) is built around a tabbed central editing view-space and a *Book Browser* to the left. You double-click files in the Book Browser with your mouse, and an editing (or display) tab is added to the main pane. These tabs can then be closed, reordered or opened again.

Main user interface presents a typical example of the UI at work.

2.1 What are all these files?

You may be wondering what are all these files in the Book Browser. Why isn't the book displayed as a set of pages? Well for that you first need to understand how the EPUB format works. A more in-depth description can be found in the *EPUB Overview* section, but what follows is a short summation (but do read that section too).

An EPUB file is just a container for various resources that go into an EPUB book. The text of the book is in XHTML files and usually, content producers use one XHTML file for one chapter of the book. You can also commonly find several CSS styling files, images and fonts files. Now, along with all these resources come several other special files that describe the Table of Contents, the book's *metadata*, the reading order of the XHTML files, list all the resources present in the container and more.

A *Reading System* (RS) takes an EPUB file and then displays to the user the first XHTML file in the specified reading order. It doesn't present it like a web browser would—as one large page with a scrollbar—but “paginates” the content into pages the size of the user's screen. When the user reaches the end of the XHTML file, the next page will be the start of the succeeding XHTML file in the reading order.

What the RS provides is merely a paged rendition of those resources. *Every* RS will provide a slightly different rendition that adapts to the screen size and technical limitations of the device it's running on; that's the whole point of EPUB.

Now, that is how an RS works—everything is displayed in pages. But this is not the ideal way to work with a book when you're creating it. Books can well go into thousands of pages.

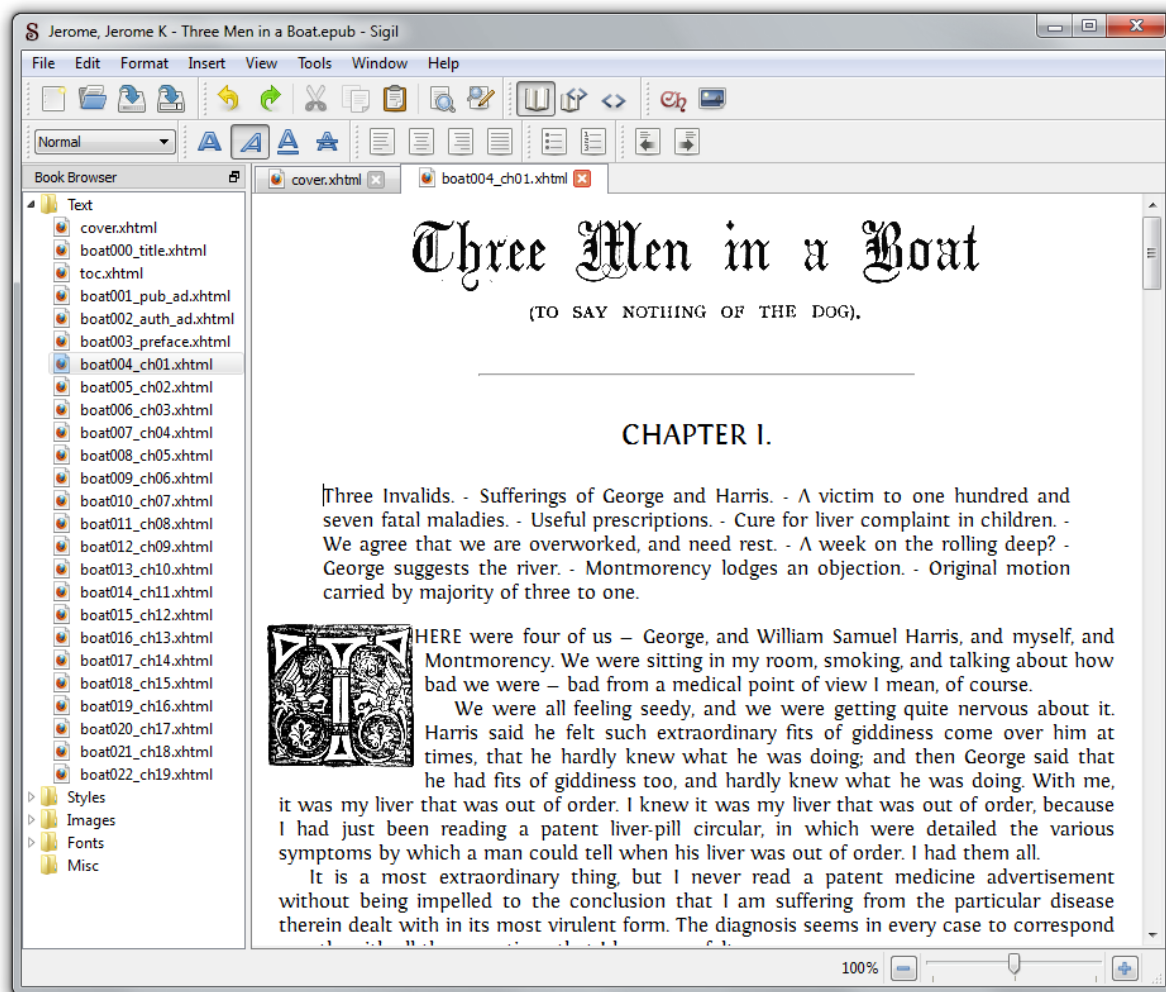


Figure 2.1: Main user interface

So Sigil presents a file-focused editing environment. You can open every XHTML file and edit it in a WYSIWYG manner in the *Book View*, or edit the code directly in the *Code View*. You can edit the CSS, XPGT and other files, and view the images one by one.

To add and edit that special information like the Table of Contents and the metadata, Sigil presents you with purpose-built editors. You can access them through the **Tools** menu. Everything you enter in them is saved and exported when the resource files are bundled together and the EPUB book is built. When you open the resulting EPUB file in some RS, it will be displayed as a paged book.

2.2 The Views

2.2.1 Book View

The Book View displays the XHTML files “rendered”, that is as they would look like in a *Reading System* when presented to the user.

Most of the WYSIWYG buttons and actions only work in this view. The **Headings** combo-box for instance changes the selected paragraph into a heading from which a Table of Contents can be built (see the *Table Of Contents Editor* section). You can also insert images, chapter breaks and apply all the standard formatting operations like bold, italic, paragraph alignment and more.

This view is only available for XHTML files.

2.2.2 Code View

The Code View provides a way to edit the underlying code of XHTML, CSS and XPGT files while displaying it with advanced syntax highlighting. Line numbers are displayed in a gutter on the left side of the pane.

An example of what an XHTML file looks like in this view can be seen in the *Code View*.

2.2.3 Split View

The Split View is a sort of combination between the previous two views: content is shown rendered in the top half, and the corresponding code is shown in the bottom half. Switching between one half to the other synchronizes their positions. The dividing bar between the bars can be dragged and adjusted with the mouse.

This view is also only available for XHTML files. An example is shown in the *Split View*.

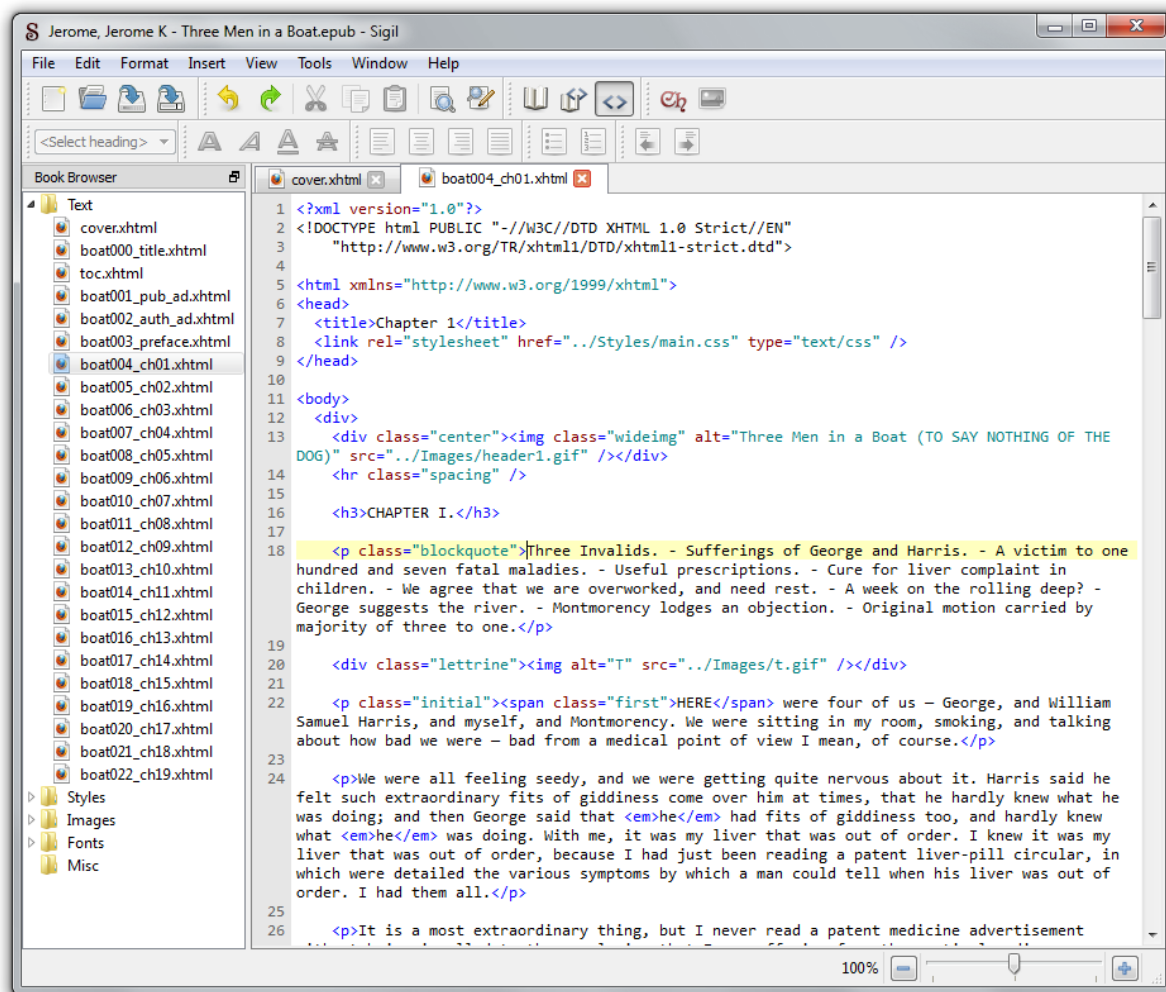


Figure 2.2: Code View

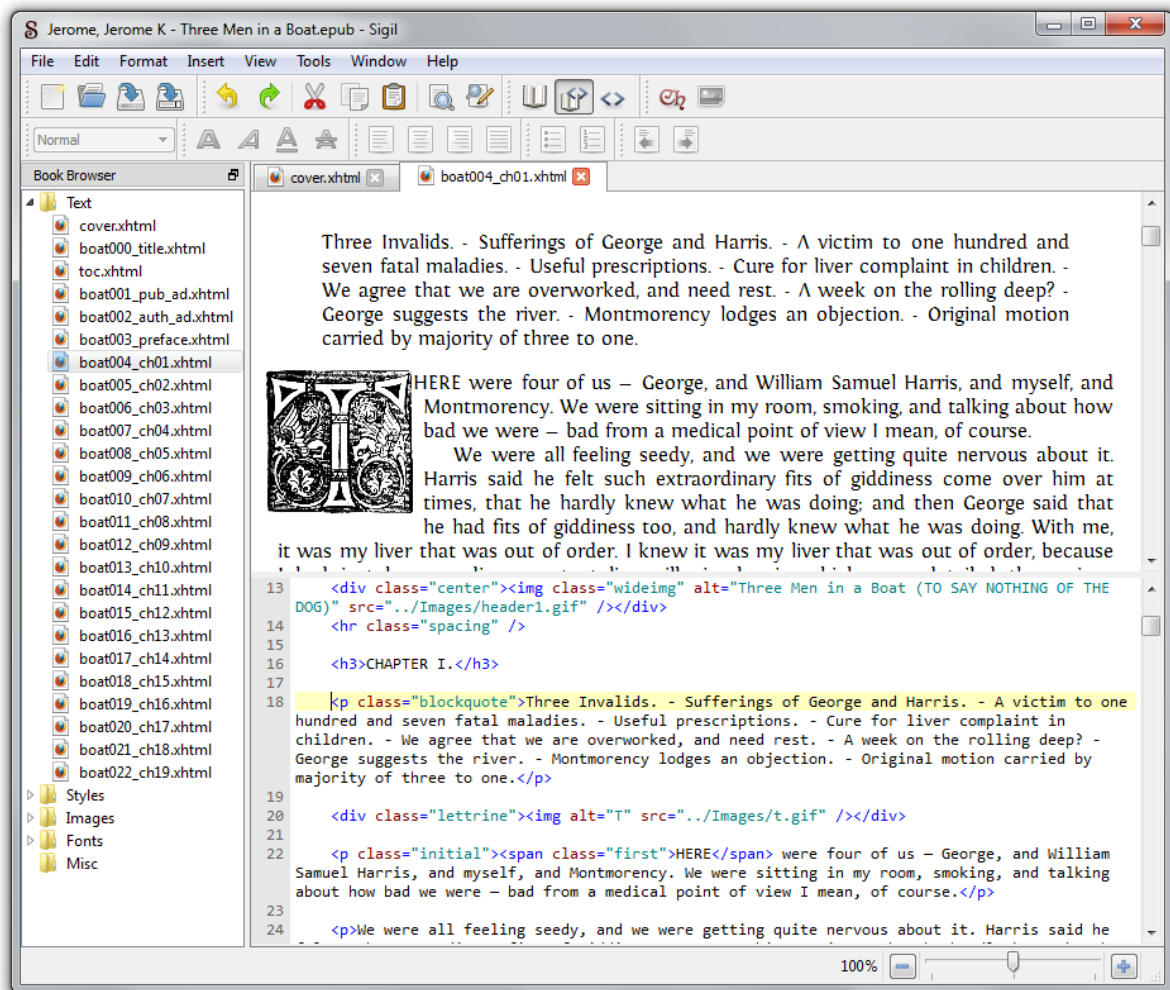


Figure 2.3: Split View

FIND & REPLACE

Sigil includes a powerful Find & Replace (F&R) dialog. By default, it opens in its collapsed state; see *Find & Replace, collapsed*. To see and modify the available options, press the **More** button. The dialog would then expand (see *Find & Replace, expanded*) and show you the following options:

Match whole word only Available only for the *Normal* search mode. Using `car` with this option selected would match `car`, but not `carpet`.

Match case All matching is by default case-insensitive. Using `John` with this option selected would match `John`, but not `john`.

Minimal matching Available only for the *Wildcard* and *Regular expression* search modes. Makes the search string match as little as possible.

Direction Up Searches from the *caret* upwards.

Direction Down Searches from the *caret* downwards.

Direction All Searches from the *caret* downwards, but loops back and starts from the beginning when the end is reached.

The **Look in** combo-box offers various places where Sigil should search. Currently, it can only search in the currently open file or all the HTML files ¹. In the future it will be able to search across all the CSS files and all the files of any type in the EPUB book.

This leads us to the various search modes, which represent the true power of the Find & Replace dialog; the *Normal* search mode just matches every character you typed exactly as it is, but the other modes are more involved.

3.1 Wildcard mode

One of the available advanced matching modes is the *Wildcard* mode. It should be familiar to anyone who's used file globbing in `bash` or Windows' `cmd`. It is fairly straightforward:

¹ And only in the *Code View*, not in the *Book View*. This limitation exists because of technical restrictions, and will be removed in time.

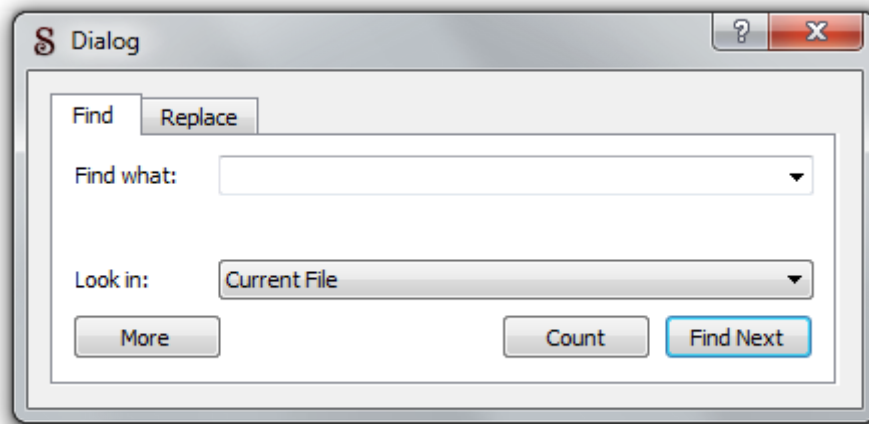


Figure 3.1: Find & Replace, collapsed

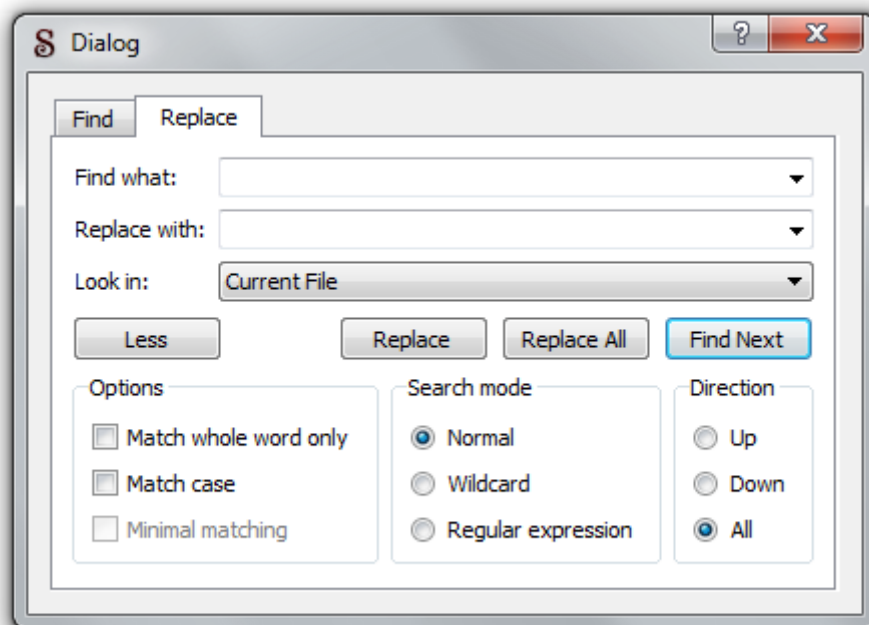


Figure 3.2: Find & Replace, expanded

Character	Matches
a	Any character represents itself by default. a would match a.
?	Matches <i>any</i> single character.
*	Matches <i>zero or more</i> of <i>any</i> characters.
[. . .]	Any character in the set; [abc] would match a <i>or</i> b <i>or</i> c.

So if for example you used the search string `[123]abc?dd*` with the *Wildcard* mode selected, it would match any of the following: `1abceddertert`, `3abc9dd--.!`, `2abc_dd!#$79840sd` `12adad` `ad` and many more.

3.2 Regular expression mode

The regular expression (*regex*) engine used in Sigil is very “Perl-like”². A general introduction to regular expressions is beyond the scope of this manual. There are many good overviews on the internet; one of which is available [here](#). That site will help you get started with regexes.

Here are *some* of the differences from Perl:

- In Sigil `^` always signifies the start of the string (apart from within character classes), so carets should be escaped unless used in this way. The same applies to `$` which in Sigil always signifies the end of the string.
- Non-greedy matching cannot be applied to individual quantifiers. Expressions like `abc*?dd` are not allowed. Use the *Minimal matching* option to set minimal matching on the whole expression.
- Back-reference syntax is *sed*-like, that is in the form of `\#`, e.g. `\1`, `\2`, `\3` etc. (`\0` is the whole matched string).
- While zero-width positive and zero-width negative lookahead assertions (in the form of `(?=pattern)` and `(?!pattern)`) are supported, Perl’s look-behind assertions, “independent” subexpressions and conditional expressions are **not** supported.

² The current regex engine used is actually Qt’s `QRegExp`. It will eventually be replaced with `PCRE` because of the latter’s advanced features and performance.

BOOK BROWSER

The Book Browser (BB) offers a view into the structure of your EPUB book. You can see all the different files that make up the archive, from *XHTML* files to images.

Hint: If your not familiar with the internal structure of an EPUB file, see the *EPUB Overview* chapter.

The Book Browser pane can be either docked, or undocked. A docked BB can be undocked by clicking on the little window icon in the top right corner, and docked by simply dragging the new window to the edge of the main Sigil window and then releasing the mouse button. An image of it undocked can be see in *Book Browser, undocked*.

You can also open and close the Book Browser through the **View** menu.

4.1 Content folders

The view is organized into several folders which can be expanded or collapsed by clicking on the little arrow icon next to them:

- Text — all the XHTML files that make up the text of the work;
- Styles — the CSS and XPGT stylesheets;
- Images — PNG, JPG, GIF and other image types you have in your book;
- Fonts — TTF and OTF fonts;
- Misc — miscellaneous files.

In most folders, the files are listed alphabetically. The exception is the `Text` folder in which the files are listed in their reading order. Since the EPUB archive consists of several XHTML files (customarily one per book chapter), the order in which they are displayed to the user needs to be determined in advance. Thus, the `Text` folder allows you to click a file and drag it amongst the other files in that folder. The file at the “top” is the one which will be shown to the user first, and the one at the bottom will be shown last.

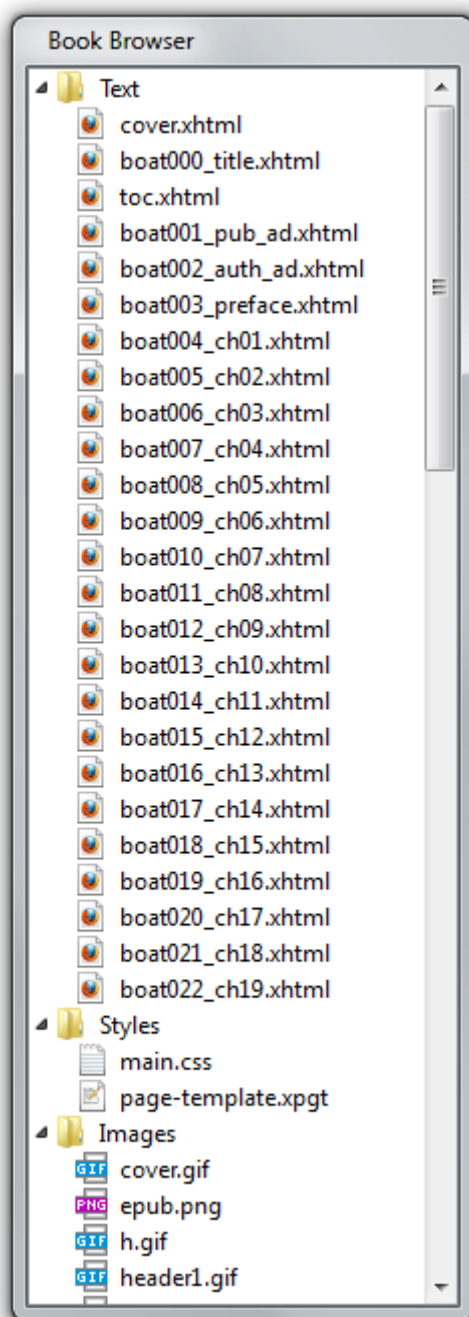


Figure 4.1: Book Browser, undocked

4.2 Context menu

Right-clicking on an item in the Book Browser brings up a context menu, and depending on the item on which you invoked the menu, you'll be able to:

- Add existing items,
- Add new items,
- Remove the file,
- Rename the file,
- Add semantic information.

If you select **Add existing items**, then an **Open file** dialog will be shown. From here, you'll be able to add any type of file that can be embedded inside an EPUB, and they'll be automatically routed to the correct folder. If you pick an HTM, HTML or XHTML file (or several), Sigil will also import all the resources that those files reference, like CSS stylesheets or images (but *not* other XHTML files). Using this technique, you can build up your EPUB file from several HTML files prepared in advance.

The **Add Semantics** sub-menu is special. It is only displayed for XHTML files and images, and the contents differ. When invoked on an XHTML file, it enables you to indicate that that file contains the book's dedication section, the glossary, the foreword, the preface and many more ¹. Some (but few) Reading Systems will then use this information and display it to the user.

For images, a different semantic action is offered. You can mark an image as a *cover*. This bit of semantic information is important for several *Reading Systems*, the most prominent of which is the *iBooks* application for the *iPad*. If an image is not marked as a cover, the book won't have a cover set in the iBooks "bookshelf".

Hint: Sigil has heuristics that will mark the appropriate image as the cover if you don't do it yourself. If the first file in the reading order is "very small" and has only one image in it ², that image will be selected as the cover.

So if you follow best practices, Sigil helps you out. Still, mark it by hand if you can. You will always know better than the machine.

¹ For those interested in the technical details, this information is stored in *The OPF file's* <guide> element.

² Sigil looks for a normal tag or an SVG <image> one.

META EDITOR

The Meta Editor is what you use when you want to add or edit metadata for your EPUB book. As the *EPUB Overview* explains, the standard allows for hundreds of different metadata entries to be added to any book.

5.1 Collapsed UI

When you first open the Meta Editor, it will be in its collapsed state, as shown in *Meta Editor, collapsed*. This allows you to add the minimum metadata elements that *all* EPUB books need to have: the **Title** of the work, the **Author** and the **Language** in which it is written. If you don't include these elements, then tools like *epubcheck* will complain that your book has errors (rightfully so).

This information is the bare minimum you should provide. You can always go the extra mile though. The expanded version of the dialog enables you to do just that.

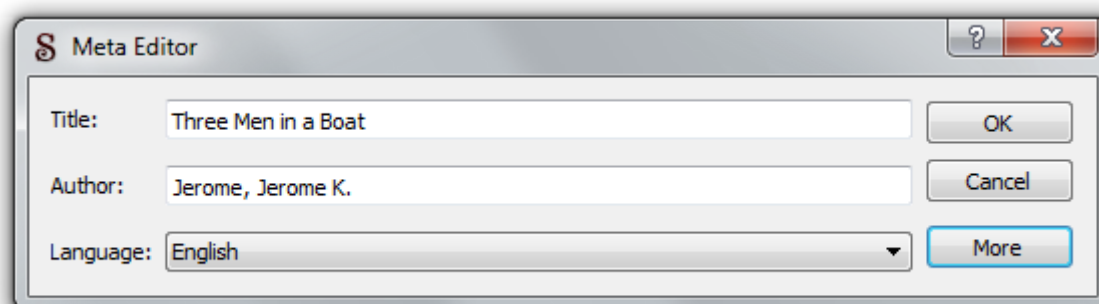


Figure 5.1: Meta Editor, collapsed

Hint: You can list several authors of a book by separating the names with a semicolon. Like this: Smith, John; Doe, Jane.

5.2 Expanded UI

If you click on the **More** button, the dialog will expand; see *Meta Editor, expanded*. You now have a table listing all the other types of metadata present in your EPUB book. You can remove an item by selecting it and clicking the **Remove** button, or add new items with the **Add Basic** and **Add Adv[anced]** buttons.

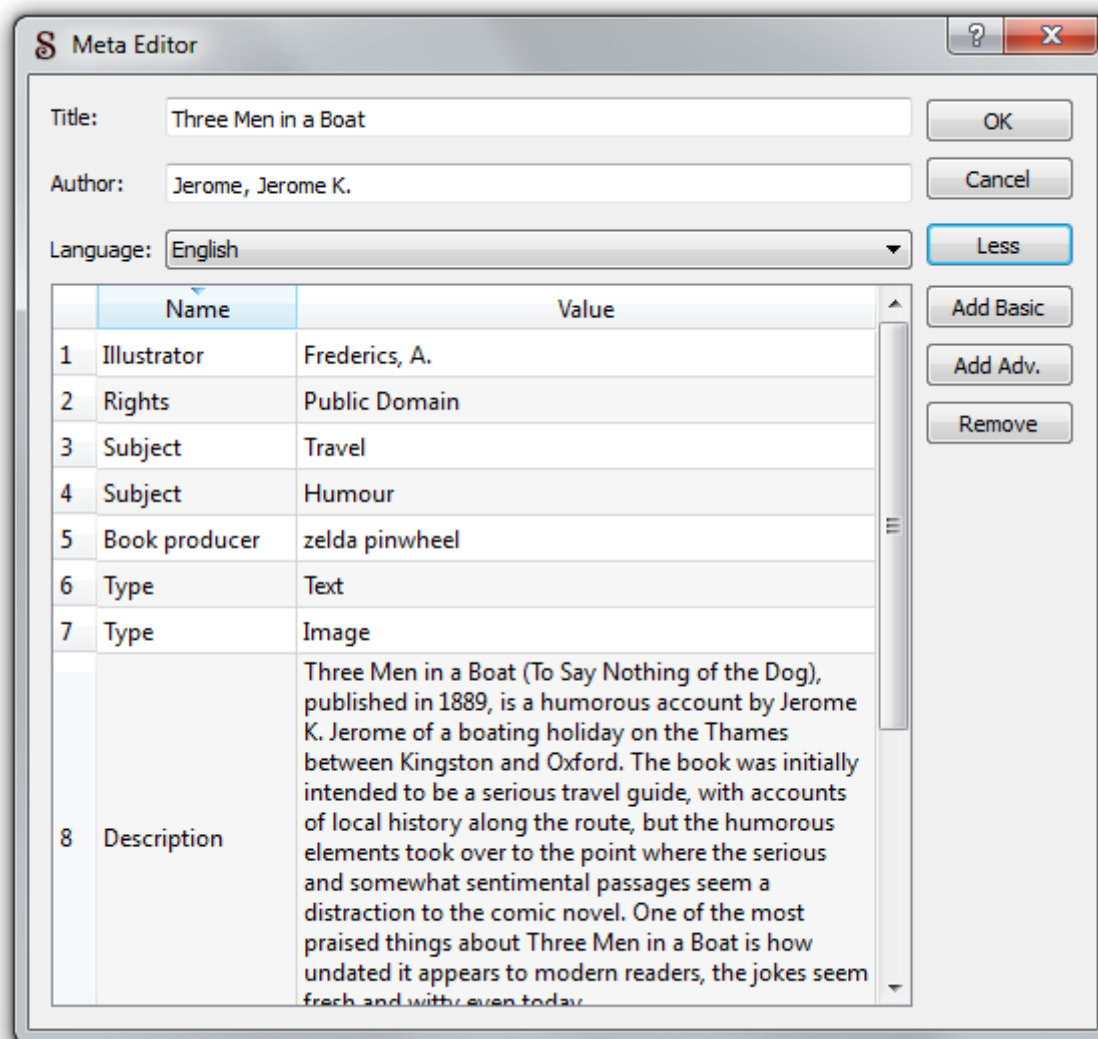


Figure 5.2: Meta Editor, expanded

The basic metadata items (see *Basic metadata list*) are things like the dates of publication, creation and modification; description; content coverage; intellectual property rights; the ISBN, ISSN and DOI numbers etc. You don't need to use any of these, the only ones that are required are the aforementioned title, author and language. But if you do know any of this information, do add it to your books.

While there are only a few basic metadata items, there's more than two hundred advanced entries (see *Advanced metadata list*), ranging from *Illustrator* to *Patent holder* to

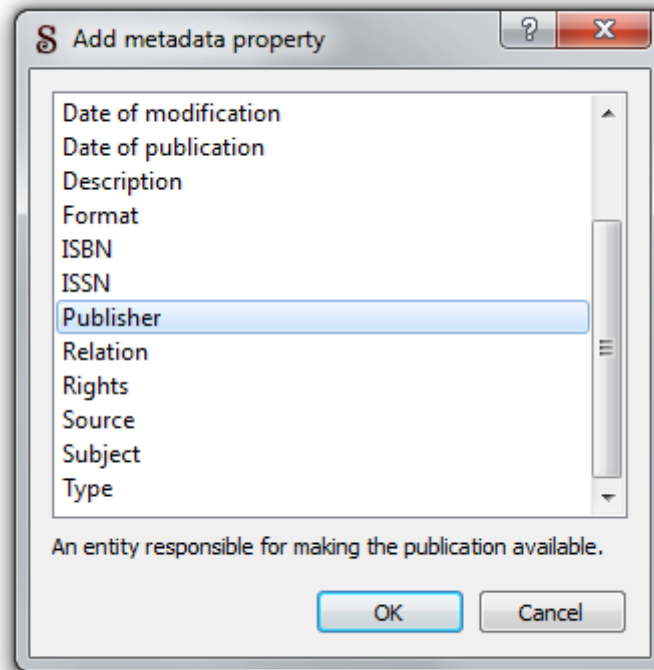


Figure 5.3: Basic metadata list

Performer. These all describe the various contributors to a work, and are terms defined and used by the Library of Congress.

Hint: Do you see the line that divides the list of metadata entries and the description text? Put your mouse cursor over it. Yes, it can be dragged to increase/decrease the amount of space for the description area.

In any field where you are expected to enter the name of a person (like *Author* or *Performer*), try to write the name in a “normalized” form. For instance, instead of John Smith, write Smith, John. Why? Because the EPUB standard provides alternate ways of storing the metadata information for people involved with the work. If you use John Smith, Sigil will store this information as just John Smith and nothing else. But if you use Smith, John (notice the comma), then the name will be stored in *two* ways. People who read your book will then see John Smith, but the *Reading System* will categorize the book under Smith, John.

This makes machine processing easier, and also makes more sense for anyone searching through their book collection.

Hint: For the technical details regarding the storage of metadata, see *The OPF file*.

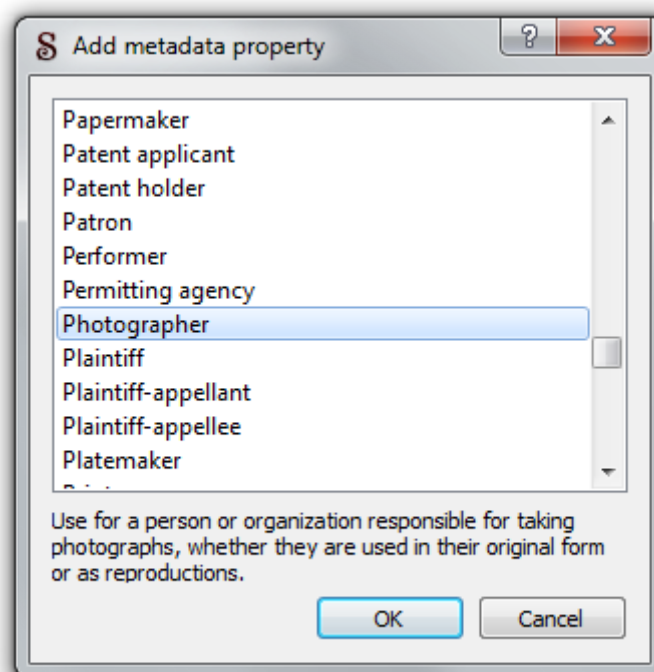


Figure 5.4: Advanced metadata list

TABLE OF CONTENTS EDITOR

Every book needs a Table of Contents (TOC). It makes navigation simpler, and also provides a quick overview of the book’s structure. With Sigil, one can create a hierarchical TOC with ease. The user interface is illustrated in *TOC editor*.

You should provide your books with a TOC, since clicking on an item in it in a *Reading System* “jumps” the user to that point in the book. This can be very useful.

6.1 Building the TOC

Sigil follows standard word-processor practice by creating a TOC from the headings used in the text. The *Book View* provides a drop-down box for changing normal text into a heading level, from one to six. Headings of a lower level (“bigger” headings) become the parents of headings of a higher level (“smaller” headings) that succeed them.

Let’s look at an example. If the headings that are used in your book are in the level order of 1 2 2 3 3 1 2 4 3, they would be ordered in the TOC like this:

```
Heading 1
  Heading 2
  Heading 2
    Heading 3
    Heading 3
Heading 1
  Heading 2
    Heading 4
    Heading 3
```

Usually, the “natural” TOC that Sigil creates using this technique will be enough for 90% of the books you’ll want to make. But you can always make changes directly.

Hint: For the technical details regarding the storage of the Table of Contents, see *The NCX file*.

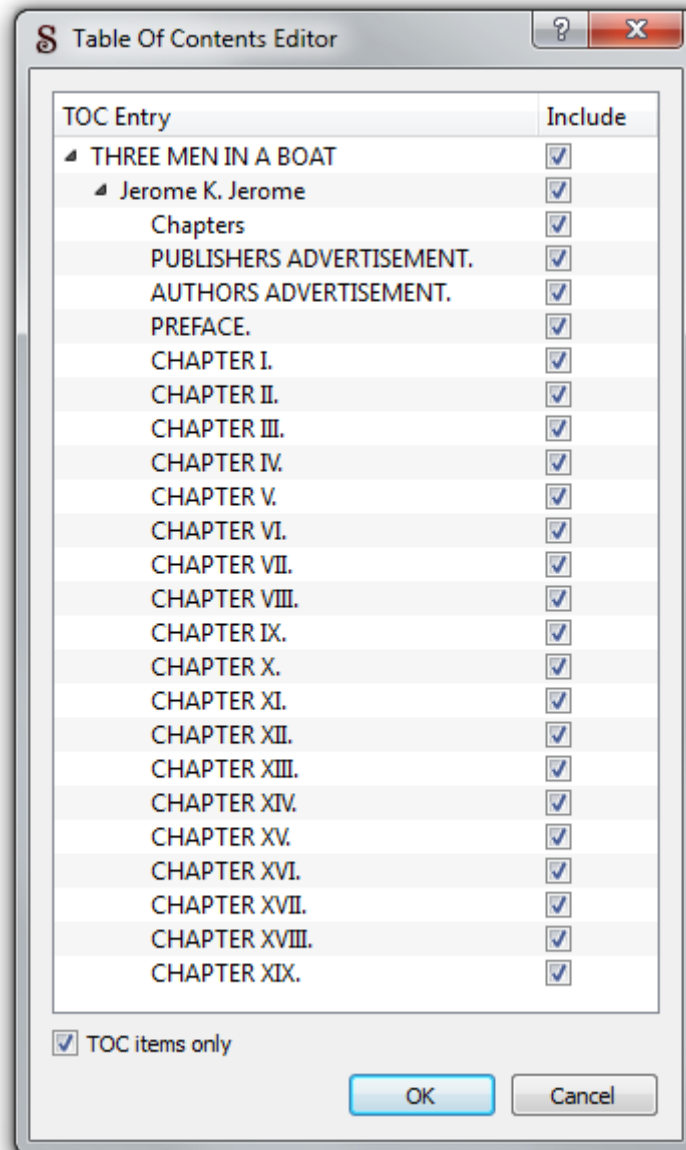


Figure 6.1: TOC editor

6.2 The editor interface

The editor shows you what the TOC will look like on a *Reading System*. Double-clicking on the text of a heading makes the text editable. Note that any changes here will also change the heading in the text of the book.

Every entry also has a checkbox next to it. Unchecking the box removes that heading from the TOC, but the heading itself remains in the text. Checking the box includes the heading back in.

Notice the **TOC items only** checkbox at the bottom of the dialog. If the option is checked, the list shows *only* the items that will be included in the final TOC: the headings that will be ignored (those that were unchecked) are not shown. Checking this *TOC items only* option shows them again.

Note: Unchecking a heading while the *TOC items only* option is selected will make it “disappear” from the list. You can see it again by deselecting the *TOC items only* option.

6.3 Advanced uses

Advanced users may want to show a different value in the TOC than what the actual heading text is. This can be easily achieved by locating the heading element in the *Code View* and adding a `title` attribute to it.

```
<h1 title="Alternate text">Normal text</h1>
```

The attribute should hold the alternate text that should be used for the TOC.

You can also use this functionality to have images as the targets of the TOC entries:

```
<h1 title="Text in TOC"></h1>
```


EPUB OVERVIEW

The EPUB format is a standard created by the *International Digital Publishing Forum* (IDPF). It consists of three separate specifications:

- *Open Publication Structure* (OPS), which describes how the content is to be presented;
- *Open Packaging Format* (OPF), which describes how the content files and resources are connected into a *logical* whole, a *publication*;
- *OEBPS Container Format* (OPS), which describes how the publication is encapsulated in a ZIP archive.

This chapter will provide a brief overview of the format aimed at beginners.

7.1 Open Publication Structure

This specification describes what the different content files of the book should look like. The basis of the content files is *XHTML* 1.1. An example document follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <link rel="stylesheet" type="text/css"
      href="../Styles/stylesheet.css" />
  </head>
  <body>
    <p>This is the first paragraph.
      This is the first paragraph.
      This is the first paragraph.
    </p>
    <p>This is the second paragraph.
      This is the second paragraph.
      This is the second paragraph.
```

```
</p>

</body>
</html>
```

You would see something very similar to this if you switched to *Code View* in the main Sigil *UI*. Notice the pattern: content is *marked up* between the start and end *tags* of an *element*. This is how *XML* (on which XHTML is based) works. The `<p>` element is a good example; it delimits the words and sentences that should be regarded as making up a single paragraph. There are many more elements in XHTML.

With XHTML, the user semantically marks up the content of the document and specifies its structure. Usually, a *CSS* stylesheet is “applied” to the document. In this example, it is linked with the `<link>` element; this tells any XHTML renderer to use this stylesheet when rendering the document.

What does the renderer do? It looks at the marked-up document and the linked stylesheets and determines how this content should be displayed on the user’s screen. The XHTML document describes *what* is the content, and the CSS stylesheet describes *how* it should be laid out and formatted. XHTML answers questions like: “What is a paragraph? What is a list of items? What is a table?”, and CSS answers questions like: “How much should the paragraphs be indented? What color is the text? What fonts should be used? How big is the text?”.

You create many XHTML documents ¹ for one EPUB file. Customarily, you create one per book chapter, but you can do it differently if you want to (but do so only if you have a good reason). For novels, one stylesheet is usually enough and is used in all the XHTML files.

Now you may be wondering, why should readers of an EPUB have to jump from one XHTML file to the next? Why can’t they just see the book as one text “flow”? The answer is that they do. The readers are *not* aware that an EPUB file is an archive of several different files since their *Reading System* always displays everything in it as one text flow. Even when every chapter is actually a separate file, when the user pages through their book, they see one chapter ending, and on the next page, the next chapter starting. They are *never* aware that their *Reading System* has actually transitioned them from one file to the next.

You may be surprised, but Microsoft Word’s new DOCX format is also several marked-up files inside a ZIP archive. And yet you see it as “one” document.

7.2 Open Packaging Format

This specification deals with connecting the various XHTML, CSS, font and image files into a “publication”. It brings in two new XML files: the OPF and the NCX.

¹ In the context of EPUB, XHTML documents are often called OPS documents.

7.2.1 The OPF file

The OPF file describes several major components:

- The metadata — the metadata for the publication;
- The manifest — lists all the files that make up the publication;
- The spine — provides a linear reading order of the XHTML files;
- The guide — provides a set of references to some of the basic structural elements of the publication, such as a table of contents, foreword, bibliography, etc.

Here is an example OPF file:

```
<?xml version="1.0"?>
<package version="2.0"
  xmlns="http://www.idpf.org/2007/opf"
  unique-identifier="BookId" >

  <metadata xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:opf="http://www.idpf.org/2007/opf">
    <dc:title>Tale of Two Cities</dc:title>
    <dc:creator opf:file-as="Dickens, Charles"
      opf:role="aut">Charles Dickens</dc:creator>
    <dc:language>en</dc:language>
    <dc:identifier id="BookId" opf:scheme="ISBN">
      123456789X
    </dc:identifier>
  </metadata>

  <manifest>
    <item id="ncx" href="toc.ncx"
      media-type="application/x-dtbnex+xml"/>
    <item id="chapter001" href="Text/chapter001.xhtml"
      media-type="application/xhtml+xml"/>
    <item id="chapter002" href="Text/chapter002.xhtml"
      media-type="application/xhtml+xml"/>
    <item id="loi" href="Text/loi.xhtml"
      media-type="application/xhtml+xml"/>
    <item id="stylesheet" href="Styles/stylesheet.css"
      media-type="text/css"/>
    <item id="cover" href="Images/cover.png"
      media-type="image/png"/>
    <item id="caecilia" href="Fonts/caecilia.otf"
      media-type="application/x-font-otf"/>
  </manifest>

  <spine toc="ncx">
    <itemref idref="chapter001" />
```

```
    <itemref idref="chapter002" />
</spine>

<guide>
  <reference type="loi" title="List Of Illustrations"
    href="loi.xhtml" />
</guide>

</package>
```

You can clearly see the metadata, manifest, spine and guide elements and their children. There really is nothing complicated about this, lots of people end up writing all this information by hand. It's tedious, but doable. Sigil writes this file automatically for you, and in the future it will provide the user with the means of editing it directly if he wants the extra power that comes with this ability.

7.2.2 The NCX file

The other file that every publication needs is the *Navigation Center eXtended* (NCX). This file describes the hierarchical Table of Contents (TOC) for the publication. Here's an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ncx PUBLIC "-//NISO//DTD ncx 2005-1//EN"
"http://www.daisy.org/z3986/2005/ncx-2005-1.dtd">

<ncx version="2005-1"
  xml:lang="en"
  xmlns="http://www.daisy.org/z3986/2005/ncx/">

  <head>
    <meta name="dtb:uid" content="123456789X"/>
    <meta name="dtb:depth" content="1"/>
    <meta name="dtb:totalPageCount" content="0"/>
    <meta name="dtb:maxPageNumber" content="0"/>
  </head>

  <docTitle>
    <text>Tale of Two Cities</text>
  </docTitle>

  <docAuthor>
    <text>Dickens, Charles</text>
  </docAuthor>

  <navMap>
```

```
<navPoint class="chapter" id="chapter001" playOrder="1">
  <navLabel><text>Chapter 1</text></navLabel>
  <content src="chapter001.xhtml"/>
</navPoint>

<navPoint class="chapter" id="chapter002" playOrder="2">
  <navLabel><text>Chapter 2</text></navLabel>
  <content src="chapter002.xhtml"/>
</navPoint>
</navMap>

</ncx>
```

The `<navPoint>` elements point either to whole documents, or to the specific elements within those documents. They can be nested to create a hierarchical TOC.

Sigil creates this file from the headings present in your XHTML documents. Every heading is referenced by a `navPoint`, and headings of different levels interact to create a hierarchy. More details of this behaviour can be found in the *Building the TOC* section. As with the OPF file, future versions of Sigil will enable direct editing of this file for those who want it.

Note: If your heading is “near” the beginning of your XHTML file, Sigil will link directly to that file and not to the heading element. This is because on some *Reading Systems*, linking directly to an element slows down the display of the TOC.

7.3 OEBPS Container Format

The OCF specification states how the publication should be packaged. Large parts of it deal with things like encryption and alternate renditions of a publication within a single archive, but the main idea to take away is that EPUB files are basically ZIP archives of the files comprising the publication. The ZIP format stores the files using the DEFLATE compression algorithm.

EPUB files also must have a META-INF folder with a `container.xml` file pointing to the OPF file of the publication. Sigil takes care of all of this for you.

GLOSSARY

caret The blinking vertical bar in the text that most people erroneously call the cursor.

CSS Cascading Style Sheets.

epubcheck A tool used to validate EPUB files. Since EPUB files need to follow the EPUB specification, this tool was made to check for basic errors in the file that would make it non-valid. Do note that it is not a panacea and it is not perfect; it doesn't check everything. Also note that even if `epubcheck` reports that a file is valid, that doesn't mean the file will render flawlessly on all *Reading Systems*. None of those are perfect either, and they all have their little quirks and idiosyncrasies.

With all of that in mind, it's still a *very* good idea to make sure your EPUB files pass `epubcheck` without errors or warnings.

`epubcheck` can be downloaded [here](#).

metadata The information about the book, like the book's title, author, publication date, publisher, subject etc.

Reading System Any combination of hardware and/or software that is used to present the EPUB book to the reader. For instance, it can be an application like *Adobe Digital Editions* or a hardware device like the *Barnes & Noble Nook*.

UI User Interface.

WYSIWYG What You See Is What You Get.

XHTML Extensible Hypertext Markup Language.

XML Extensible Markup Language.

XPGT XML Page Template.