HTML5 as an alternative to DITA and DocBook

Hussein Shafie

XMLmind Software

October 15, 2020

Table of Contents

Problem and solution 1
What is XMLmind Ebook Compiler? 3
XMLmind Ebook Compiler primer 4
Give it a try 14

Problem and solution

The problem

You have to write the documentation of an advanced technical product (software, hardware, service, etc) and you are not sure which technology or tool is the best choice for doing this.

You have already excluded the idea of using a word processor such as Microsoft Word because this documentation, besides being expected to be very large (several hundreds of pages long, hundreds of cross-references, dozens of tables and figures, an extensive index, etc), is mainly intended to be published online as a set of HTML pages.

You have of course heard about DITA and DocBook and about XML Editors and Content Management Systems having built-in support for these technologies. However these XML vocabularies are so large and so complex that you are already discouraged. Moreover you have heard that converting DITA or DocBook documents to deliverables looking right requires you to delve at best, into XSL, and at worst, into the arcanes of advanced conversion toolkits[1].

The HTML solution

The most important format for your deliverables being HTML, why not directly write your technical documentation in HTML and style it using CSS?

At first this seems to be a great idea but you must realize, even with the help of a good HTML editor, you'll lack many of the features provided by DITA or DocBook and their conversion toolkits:

- Automatic generation of global and local table of contents.
- Automatic generation of a “back-of-the-book index”.
- Automatic numbering of parts, chapters, appendices, sections, figures, tables, examples and equations.
- Automatic creation of links between some of the book divisions.
- Automatic generation of text in cross-references.
- Footnote support.
- Conditional processing (also called *profiling*).
- Inclusion of “boilerplate text” (a note, the product name, a version number, etc) at different locations of the book.

In fact, this HTML approach can work but you need more than an HTML editor for that. You need a tool letting you create and publish full books —not just pages— in HTML. Some of these tools are:

- MadCap Flare, a desktop application which seems to use a variant of XHTML as its native source format. Commercial.
- O’Reilly Atlas, a web-based platform which uses HTMLBook (XHTML5 based) as its native source format. Not sure if it's commercial or reserved to O’Reilly authors.
- Pressbooks, a web-based platform. Commercial. Free to use with "Pressbooks" watermarks added to the deliverables.
- Our own, free, open source, XMLmind Ebook Compiler.

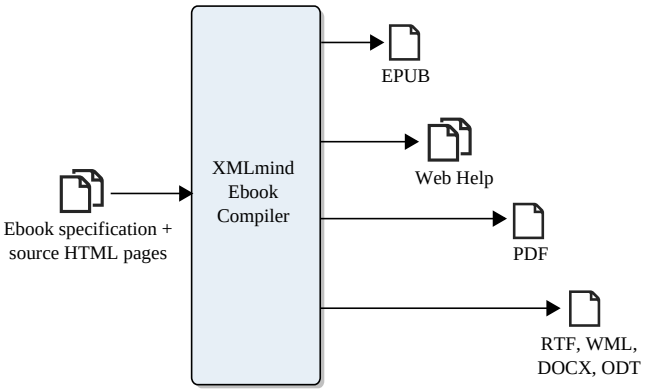
We'll now explain

[1]
- DITA Open Toolkit.
- XMLmind DITA Converter, a serious alternative to the DITA Open Toolkit.
- DocBook XSL stylesheets.

- what exactly is XMLmind Ebook Compiler (**ebookc** for short),
- how **ebookc** can be used to author and publish technical manuals written in HTML5.

What is XMLmind Ebook Compiler?

XMLmind Ebook Compiler (**ebookc** for short) is a free, open source tool which can turn a set of HTML pages into a self-contained *ebook*[2]. Supported output formats are: EPUB, *Web Help*, PDF[3], RTF, WML, DOCX (MS-Word) and ODT (OpenOffice/LibreOffice)[4].



You can of course use **ebookc** to create books having a simple structure like novels, but this tool also has all the features needed to create large, complex, reference manuals:

- Builds on topic-oriented structuring like DITA or DocBook 5.1. (Each source HTML page is expected to deal with a single topic.)
- Automatic generation of global and local table of contents.
- Automatic generation of a “back-of-the-book index”.
- Automatic numbering of parts, chapters, appendices, sections, figures, tables, examples and equations.
- Automatic creation of links between some user-specified book divisions.
- Automatic generation of text in cross-references.
- Footnote support.
- Conditional processing (also called *profiling*).
- Built-in support of XInclude (allows reuse of content at different locations in the book).

Being based on HTML, **ebookc** relies on CSS to create nicely formatted books and this, even for output formats like PDF and DOCX which are not directly related to HTML and CSS.

[2] Here “ebook” shall be understood in the widest possible sense.

[3] Requires an XSL-FO processor like Apache FOP, RenderX XEP, Antenna House Formatter to be installed and registered with XMLmind Ebook Compiler (for example, using option `-foconverter`). We'll assume in this manual that you have downloaded and installed the distribution of XMLmind Ebook Compiler which includes Apache FOP.

[4] Requires XMLmind XSL-FO Converter to be installed and registered with XMLmind Ebook Compiler (using option `-xfc`).

XMLmind Ebook Compiler primer

A book is an assembly of HTML pages

The basic idea is simple. You author a set of HTML pages and then you create an ebook specification assigning a role —part, chapter, section, appendix, etc— to each page. Example: `primer/book1.ebook`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      href="titlepage.html">
  <frontmatter>
    <toc/>
  </frontmatter>

  <chapter href="ch1.html"/>

  <chapter href="ch2.html"/>

  <appendix href="a1.html"/>
</book>
```

The HTML pages comprising a book may contain anything you want including CSS styles and links between the pages (e.g. `<a href="ch2.html#fig1">`). However make sure that this content is *valid XHTML*[5].

Once the ebook specification has been created, you can compile it using XMLmind Ebook Compiler and generate EPUB, *Web Help*, PDF[6], RTF, ODT, DOCX[7], etc. Examples:

```
ebookc book1.ebook out/book1.epub

ebookc book1.ebook out/book1.pdf
```

“Rich”, numbered, chapter titles

If you look at `out/book1.pdf`, you'll see that chapter and appendix titles are numbered and that these titles are copied verbatim from the `html/head/title` of the corresponding input HTML page.

It's of course possible to specify how book components should be numbered (if at all). It's also possible to replace the plain text titles of chapters and appendices by “rich” titles[8] by adding `ebook:head` child elements to the book divisions. Example: `primer/book2.ebook`:

[5] Preferably *valid XHTML5*, because `ebookc` anyway generates XHTML5 markup. “Plain HTML” cannot be parsed by `ebookc`.

[6] Requires an XSL-FO processor like Apache FOP, RenderX XEP, Antenna House Formatter to be installed and registered with XMLmind Ebook Compiler (for example, using option `-foconverter`). We'll assume in this manual that you have downloaded and installed the distribution of XMLmind Ebook Compiler which includes Apache FOP.

[7] Requires XMLmind XSL-FO Converter to be installed and registered with XMLmind Ebook Compiler (using option `-xfc`).

[8] That is, possibly containing the same elements as an HTML `p` (`em`, `kbd`, `img`, etc.)

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      href="titlepage.html" appendixnumber="A%1.">
  <frontmatter>
    <toc/>
  </frontmatter>

  <chapter href="ch1.html"/>

  <chapter href="ch2.html">
    <head>
      <title>“<html:em>Rich</html:em>” title of
      second chapter</title>
    </head>
  </chapter>

  <appendix href="a1.html"/>
</book>
```

The content of a `ebook:head` element specified this way is added to the `html/head` of the corresponding output HTML page, except for the `ebook:title` element which replaces `html/head/title`.

Assembling a book division rather than referencing an external file

We have already seen that it's possible to add a `ebook:head` child to elements like `book`[9], `chapter`, `appendix`, etc. Likewise, it's also possible to add a `ebook:body` child to any book division. Example: `primer/book3.ebook`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      appendixnumber="A%1">
  <head>
    <title>Title of this sample book</title>
  </head>
  <body>
    <content href="titlepage.html"/>
  </body>

  <frontmatter>
    <toc/>
  </frontmatter>

  <chapter href="ch1.html"/>

  <chapter href="ch2.html">
```

[9] In that matter, the root `book` element is no different from `part`, `chapter`, `appendix`, `section`, etc.

```
    <head>
      <title>“<html:em>Rich</html:em>” title of
      second chapter</title>
    </head>
  </chapter>

  <appendix href="a1.html"/>
</book>
```

In the above example, the content of the `html/body` element of file `titlepage.html` is “pulled” and added to the book. Several `ebook:content` child elements are allowed in an `ebook:body` element.

Controlling generated page names

When you generate multi-page HTML (e.g. Web Help) out of an ebook specification, it may be important to specify the names of the generated pages. It may also be useful to group several consecutive book divisions into the same output page.

This is specified using the `pagename` and `samepage` attributes of any book division. Example: `primer/book4.ebook`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      appendixnumber="A%1">
  <head>
    <title>Title of this sample book</title>
  </head>
  <body>
    <content href="titlepage.html"/>
  </body>

  <frontmatter>
    <toc/>
    <section href="intro.html" pagename="the introduction"/>
  </frontmatter>

  <chapter href="ch1.html">
    <section href="s1.html">
      <section href="s2.html" samepage="true"/>
    </section>
  </chapter>

  <chapter href="ch2.html">
    <head>
      <title>“<html:em>Rich</html:em>” title of
      second chapter</title>
    </head>
  </chapter>
```

```
28
29    <appendix href="a1.html"/>
30  </book>
```

By default, each book division is created in its own file and the name of this file comes the `href` attribute of the book division. Web Help example:

```
ebookc -f webhelp book4.ebook out/book4
```

- Without attribute `pagename="the introduction"`, the introduction would have been generated in file `out/book4/intro.html`. With this attribute, the introduction is generated in file "`out/book4/the introduction.html`".
- Without attribute `samepage="true"`, the second section would have been generated in its own file `out/book4/s2.html`. With this attribute, the second section is appended to file `out/book4/s1.html`, also containing first section.

But wait a minute… HTML has not enough elements to write books

That's right, some semantic elements like admonitions, footnotes, etc, found in larger XML vocabularies like DITA or DocBook are missing from XHTML5. However, it's easy to emulate these missing elements by defining semantic values for the `class` attribute of standard HTML elements (typically `span` and `div`).

XMLmind Ebook Compiler has special support for the following semantic class names:

Semantic class	Description
`<figure class="role-equation">`	A “displayed equation” having a title (`figcaption`).
`<figure class="role-example">`	An example —for example a code snippet— having a title (`figcaption`).
`<pre class="role-listing-c-1">`	A code listing, possibly featuring line numbering and syntax coloring (class name suffix "`-c-1`" means: C language, first line number is 1).
`<blockquote class="role-note">`	Admonitions. Supported class names are: `role-note`, `role-attention`, `role-caution`, `role-danger`, `role-fastpath`, `role-important`, `role-notice`, `role-remember`, `role-restriction`, `role-tip`, `role-trouble`, `role-warning`.
`<span class="role-footnote">`	A short footnote, inline with the rest of the text.
`<a class="role-footnote-ref" href="#fn1">`	A call to footnote "`fn1`".
`<div class="role-footnote" id="fn1">`	Footnote "`fn1`".
`<a class="role-index-term">Cat</a>`	An index term. May be much more elaborate than the very simple example shown here.

Excerpts from file `primer/semantic_classes.html` which has been added to `primer/book5.ebook` as its second appendix:

```
...
<figure class="role-equation">
  <figcaption>Figure containing
  an equation</figcaption>
  <div>
    <math display="block"
          xmlns="http://www.w3.org/1998/Math/MathML">
      <mrow>
        <mi>E</mi>
        <mo>=</mo>
        <mrow>
          <mi>m</mi>
          <mo>#</mo>
          <msup>
            <mi>c</mi>
            <mn>2</mn>
          </msup>
        </mrow>
      </mrow>
    </math>
  </div>
</figure>
...
<p>Short footnote<span class="role-footnote">Content of
short footnote.</span>.
...
<p>Simplest index term<a class="role-index-term">Cat</a>.
Other index term<a class="role-index-term">Cat<span
class="role-term">Siamese</span></a>...</p>
...
```

Because `primer/semantic_classes.html` contains figures, tables and index terms, the following book divisions have also been added to `primer/book5.ebook`:

```
...
  <frontmatter>
    <toc/>
    <lof/>
    <lot/>
    <lox/>
    <loe/>
    <section href="intro.html" pagename="the introduction"/>
...
  <backmatter>
```

```
    <index/>
  </backmatter>
...
```

`<lof/>` specifies that a List of Figures is to be generated as a front matter. `<lot/>` means: List of Tables. `<lox/>` means: List of Examples. `<loe/>` means: List of Equations.

Nicely formatted books

If you compile `primer/book5.ebook`, you'll get a *very dull* result whatever the output format:

```
ebookc -f webhelp book5.ebook out/book5

ebookc book5.ebook out/book5.pdf
```

This is caused by the fact that all the source HTML pages referenced by `book5.ebook` do not specify any CSS style.

It's a good practice to keep it this way because this allows separation of presentation and content. However, you'll want to create nice books, so the simplest and cleanest is to add CSS styles to the ebook specification (and not to each input HTML page).

If you do it like this:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      appendixnumber="A%1">
  <head>
    <title>Title of this sample book</title>
    <html:link href="css/styles.css" rel="stylesheet"
               type="text/css"/>
  </head>
  ...
```

The above specification would *not* work because only the title page would get styled.

You need to use a `headcommon` element for that. The child elements of `headcommon` are automatically copied the `html/head` of all output HTML pages. Excerpts from `primer/book6.ebook`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      appendixnumber="A%1">
  <headcommon>
    <html:link href="css/styles.css" rel="stylesheet"
               type="text/css"/>
  </headcommon>

  <head>
    <title>Title of this sample book</title>
```

```
    <html:style>
div.role-book-title-div {
    text-align: center;
}

h1.role-book-title {
    margin: 4em 0;
    padding-bottom: 0;
    border-bottom-style: none;
}
    </html:style>
  </head>
  ...
```

In the above example:

- Element `ebook:head` may contain, not only `ebook:title`, but also any of the HTML elements allowed in `html/head`, namely `style`, `script`, `meta`, `link`. This facility is used here to give a specific style to the title page.
- Unlike `<blockquote class="role-note">` for example, which is found in the source HTML page, `<div class="role-book-title-div">` and `<h1 class="role-book-title">` are elements *generated* by XMLmind Ebook Compiler.

  Knowing about these elements is required to be able to give nice looks to the generated book. These elements and their class names are all listed in `template/skeleton.css`, with suggested CSS styles for some of these elements.

Leveraging `base.css`, the stock CSS stylesheet

As of version 1.4, the easiest way to add CSS styles to an ebook specification is to set attribute `includebasestylesheet` of element `book` to "`true`". This very simple setting guarantees to effortlessly create a nicely formatted book.

More precisely , attribute `includebasestylesheet="true"` instructs `ebookc` to include the *`ebookc_install_dir`*`/xsl/common/resources/base.css` stock CSS stylesheet in all the output HTML pages.

In the following example, we not only use `base.css`, but we also customize most of its colors by including a custom stylesheet called `custom_colors.css`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      includebasestylesheet="true">
  <headcommon>
    <html:link href="custom_colors.css" rel="stylesheet"
               type="text/css"/>
  </headcommon>
  ...
```

A sample color customization stylesheet is found in `template/custom_colors.css`.

What about output formats like PDF, RTF, DOCX?

The CSS styles specified in the ebook specification and in the source HTML pages are also used when generating output formats like PDF, RTF, DOCX, even if these formats are not directly related to HTML and CSS.

However in this case, CSS 2.1 support is partial. While there are no restrictions related to the use of CSS selectors, only the most basic CSS properties are supported. For example, generated content (e.g. `:before`) and floats are not supported at all.

There are two ways to work around this limitation:

1. Use simpler CSS styles when targeting output formats like PDF, RTF, DOCX. This is done using `@media screen` and `@media print`[10] rules. This is done in `primer/css/styles.css`:

```
blockquote.role-warning {
    font-size: 12px;
    background-color: #e1f5fe;
    color: #0288d1;
    padding: 12px 24px 12px 60px;
    margin: 16px 0;
}

blockquote.role-warning:before {
    float: left;
    content: url(star.svg);
    width: 16px;
    height: 16px;
    margin-left: -36px;
}

@media print {
    /* Floating generated content not supported.
       No need to leave room for the admonition icon. */
    blockquote.role-warning {
        padding-left: 24px;
        border-left: solid 5px #0288d1;
    }
}
```

[10] It's also possible to use `@media` *`XSL_FO_PROCESSOR_NAME`* rules, where *`XSL_FO_PROCESSOR_NAME`* is `FOP` (Apache FOP), `XEP` (RenderX XEP), `AHF` (Antenna House Formatter) or `XFC` (XMLmind XSL-FO Converter).

2. Some features like watermark images or admonition icons are directly implemented the XSLT stylesheets which generate XSL-FO[11]. Example:

```
ebookc -p use-note-icon yes book6.ebook out/book6.pdf

ebookc -f webhelp book6.ebook out/book6
```

   Without XSLT stylesheet parameter `use-note-icon=yes`, admonitions in `out/book6.pdf` would have no icons.

   Such parameter is not needed when generating Web Help (like EPUB, an HTML+CSS-based output format) because admonition icons are specified in CSS stylesheet `primer/css/styles.css`.

Creating links between book divisions

An book is specified as an assembly of source HTML pages. If you want to reuse some of these HTML pages to author other books, it is recommended to avoid creating links (e.g. `<a href="ch2.html#fig1">`) between these pages.

Fortunately, there is a simple way to create links between book divisions, which is using the `ebook:related` element. Excerpts from `primer/book7.ebook`:

```
...
<chapter href="ch1.html" xml:id="ch1">
  <related ids="ch1 ch2 a1" relation="See also"/>

  <section href="s1.html">
    <section href="s2.html" samepage="true"/>
  </section>
</chapter>

<chapter href="ch2.html" xml:id="ch2">
  <head>
    <title>“<html:em>Rich</html:em>” title of
    second chapter</title>
  </head>

  <related ids="ch1 ch2 a1" relation="See also"/>
</chapter>

<appendix href="a1.html" xml:id="a1">
  <related ids="ch1 ch2 a1" relation="See also"/>
</appendix>
...
```

See links automatically generated in first chapter, second chapter and first appendix by running for example:

[11] A standard, intermediate page-layout format which is then used by XSL-FO processors like Apache FOP or XMLmind XSL-FO Converter to generate PDF, RTF, DOCX, etc.

```
ebookc -f webhelp book7.ebook out/book7
```

Conditionally excluding some content from the generated book

This feature called *conditional processing* or *profiling* has many uses, the most basic one being to include or exclude some content depending on the chosen output format. For example, some source HTML pages may contain interactive content (e.g. a feedback form) and this interactive content simply cannot be rendered in PDF or DOCX.

In order to conditionally exclude some content from the generated book, you must first "mark" the conditional sections using `data-*` attributes. Excerpts from `primer/book8.ebook`:

```
...
<backmatter data-output-format="docx odt pdf rtf wml">
  <index/>
</backmatter>
...
```

Excerpts from `primer/intro.html`:

```
...
<blockquote class="role-tip"
            data-output-format="epub html webhelp">
  <p>This document is also available in PDF ... format.</p>
</blockquote>
...
```

You may specify one or more conditional processing `data-*` attribute on any element. Choose the attribute names you want. Such conditional processing `data-*` attribute may contain one or more values separated by space characters. Choose the attribute values you want.

If you generate a single HTML page by running:

```
ebookc book8.ebook out/book8_no_profile.html
```

the marked sections will *not* be excluded because XMLmind Ebook Compiler does not associate any special meaning to attribute `data-output-format`. However if you run:

```
ebookc -p profile.output-format html book8.ebook out/book8.html
```

then file `out/book8.html` will not have an index. Option "`-p profile.output-format html`" reads as: unless an element has no `data-output-format` attribute or has a `data-output-format` attribute containing "`html`", exclude this element from the generated content.

If you run:

```
ebookc -p profile.output-format pdf book8.ebook out/book8.pdf
```

then the introduction will not contain the tip about the availability of the document in PDF format.

Give it a try

All in all, **ebookc** is an authoring and publishing tool *nearly as powerful* as DITA or DocBook and their advanced conversion toolkits, but being based on HTML and on CSS, it is *much easier to learn*, use and customize. Moreover you can create with it ebooks which are more interactive (audio, video, slide shows, multiple-choice questions, etc) than those created using DITA or DocBook.

If the above primer seems convincing to you then you should really give **ebookc** a serious try before attempting to adopt DITA or DocBook. Download **ebookc** from this page.

Alternatively give it a try using XMLmind XML Editor Personal Edition

XMLmind XML Editor has out of the box, extensive support for creating an ebook specification and its source HTML pages and for converting an ebook specification to a number of output formats. XMLmind XML Editor Personal Edition is *free to use* by many persons and organizations.