LaTeX to ePub

Rishi T.

Abstract

We have developed a workflow to generate ePub [1] from a LaTeX document. This workflow has two main parts. The first part converts the document sources in LaTeX format to XML. We have been using this part of the workflow for many years. The second part generates ePub from the XML documents thus created.

This workflow is completely automated and makes use of TeX4ht, XSLT and ANT scripts.

1 Evolution of our XML-to-ePub workflow

At River Valley we have been engaged in the task of perfecting a workflow for the generation of high quality ePub directly from XML sources. Since we are primarily dealing with scientific, technical, and medical (STM) books with complex mathematical formulae, the original sources of these contents will invariably have been authored in TeX. As our ePub workflow demands XML as its input, we use TeX4ht to convert the TeX sources into XML.

Two years back, we developed an XML to ePub filter, but before long, we were forced to abandon it as it suffered from several deficiencies owing to its poor design and use of inappropriate technologies for processing XML sources. One of the major handicaps of this filter was that it required repeated manual intervention to edit the XML sources to suit its rigid input format. This experience forced us to review the design of the filter from the ground up, and develop a new one flexible enough to meet the needs of the evolving ePub specifications, and to be customizable enough for processing the XMLs of different DTDs. As the future of publishing seems to be moving more and more towards ePub, we thought it appropriate to invest more time and effort on it. Now the development team is happy that, at last, it can provide a robust solution.

The latest workflow is mainly based on XSLT [2] and ANT scripts [3]. Our main concerns about the workflow were the following.

- It should be user-friendly.
- Even a novice developer should be able to maintain it.
- It should be highly customisable without modifying the core area.
- It should require no manual intervention.
- It must be an XML-based and cross-platform solution.

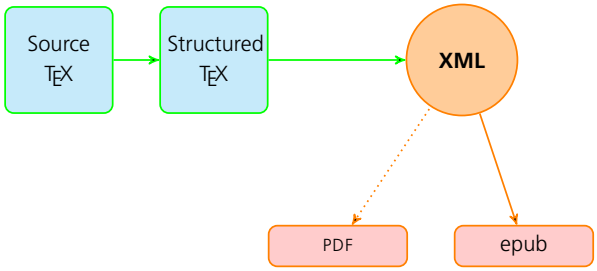

Figure 1: Schematic diagram of our workflow

2 Workflow

A simple schematic diagram of our workflow is given in figure 1. It can be described as follows.

1. Create a structured TeX document from the author's source document. Structured TeX means a TeX document, where the details are tagged clearly. An example of how author details are coded is given below:

```
\author{%
  \fnm{Rajagopal}
  \snm{CV}
}
\address{%
  \orgname{River Valley Technologies}
  \city{Trivandrum}
  \cnty{India}
}
```

   Structuring is done with the aid of TeX4ht and some scripts written in Vim.
2. This structured LaTeX document is converted to an XML format, which follows Elsevier's book DTD (`book521.dtd`).[1] TeX4ht is used for the TeX-to-XML conversion.
3. Next, bitmapped equations (images of equations) are created for all MathML tags. Images are used rather than MathML tags since current e-book readers do not support MathML rendering.
4. Then the XSLT style sheet is applied on this XML document and an ePub is created.

2.1 Working method

2.1.1 Input

(1) XML files, bitmapped equations, and any external entities (such as figures) loaded in the XML files. (2) A hub file. This is an XML file that includes the metadata and the list of XML files that should be

[1] In our experience, this is one of the best DTDs, covering almost all types of STM content as far as a standard book is concerned.

```
<files>
<title>Field Guide</title>
<author>Yakov</author>
<cover name="cover/cover.jpg"/>
<stylesheet name="epub-stuff/fg-spie.css"/>

<folder name="fg21"/>
<color fcolor="#238acb;" rcolor="#002395;"/>

<prelims>
  <file name="prelims/cover.xhtml"/>
  <file name="prelims/half-title-page.xhtml"/>
</prelims>

  <file name="spiebk-fg21-b01.xml"/>
  <file name="..."/>
  <file name="spiebk-fg21-r01.xml"/>
  ...
</files>
```

Figure 2: An example `hub.xml`

converted to ePub. The files are listed in the same order as they should be in the ePub. An example `hub.xml` is shown in Figure 2.

The source files are kept in another folder inside the working folder. In general, that folder has the same name as the project for which the ePub is to be generated.

All the source files can be either copied to the project folder or can be in different subfolders inside it. For example, one may create subfolders with chapter numbers and copy the figures and bitmapped equations of that particular chapter to that folder.

2.1.2 Process

To make the process simpler, we use the (GNU) `make`, a utility which executes commands grouped under a specific target in a file called `makefile` or `Makefile`. Separate targets are declared for each function. A single target that carries out the whole process is also available in the `Makefile`. For example,

```
make epub
```

will create an ePub, validate it and display an error log if there are any errors.

The resources of an ePub consist mainly of XHTMLs, graphic objects and several other auxiliary files. During debugging, if we have made any changes in the XHTML files directly, we need again to zip the files into an ePub format, and for this we run the command:

```
make zip
```

A complete list of our `Makefile` targets is in Table 1.

Target	Action
`file`	opens `makefile.in` to input the project id for which we need to create an ePub
`epub`	creates the ePub
`hub`	opens the hub file
`zip`	zips the files in an ePub format, assuming that all the files required for an ePub are available
`check`	validates the ePub
`renumber`	renumbers the IDs
`err`	opens the error log
`view`	opens the ePub in Lucidor (an ePub viewer)
`ncx`	opens `toc.ncx`
`opf`	opens `content.opf`

Table 1: List of targets in our `Makefile`

2.1.3 Files

The files `toc.ncx`, `content.opf` etc. mentioned in the table are generated through the XSLT style sheet. Some log files for debugging will also be generated.

3 Features

TEX to ePub through XML. The source file is a TEX file. This is converted into an XML file through an automated conversion process. The XML file generated conforms to Elsevier's book DTD. Since the primary source is TEX, TEX4ht [5] is used for TEX-to-XML conversion. During this process, one gets numerous opportunities to appreciate the power of TEX4ht and its highly configurable features for processing complex TEX documents into XML.

Conversion using XSLT. XSLT is the style sheet language recommended for XML and this is a declarative language used for the conversion of XML documents. We carry out the conversion to the ePub format from the XML using XSLT.

Minimal use of images. Except for complex math formulae, all the in-line math formulae are represented in ePubs using their HTML equivalents. For example, we can handle H_2SO_4, $E = mc^2$ etc., in HTML, whereas $\sqrt[n]{24}$ and similar formulae that do not have equivalent HTML are set as images.

Use of `dvipng`. For creating images of complex in-line and multi-line formulae, we use the application `dvipng` [6], and the images created look as beautiful as they are in the DVI.

Importing XHTML files. The real data for the ePub file comes directly from XML. However, if one has any other information (e.g., copyright pages, advertisements, call for papers etc.) which cannot be coded as XML due to DTD constraints, they can be used to create equivalent XHTML files and import them directly.

Compatibility. We have tried our best to create ePubs that are compatible with all e-book readers such as iPad, NOOK, Lucidor, Firefox etc.

Cross-platform solution. Since the conversion process uses TEX and XML technologies only, we can very well claim that this is a cross-platform solution.

4 Challenges

Making the ePub compatible with different e-book readers posed some challenges.

References

[1] http://en.wikipedia.org/wiki/EPUB
[2] http://en.wikipedia.org/wiki/XSLT
[3] http://ant.apache.org
[4] http://en.wikipedia.org/wiki/XML
[5] http://en.wikipedia.org/wiki/TeX4ht
[6] http://sourceforge.net/projects/dvipng

◇ Rishi T.
River Valley Technologies
rishi (at) river-valley dot com
http://www.river-valley.com