Electronic Documents

Wikipublisher: A Web-based system to make online and print versions of the same content

John Rankin

Abstract

Web pages and print documents exist as two solitudes: information created as a Web page may print poorly; information created as a print document may translate into an unappealing Web page. The Wikipublisher system lets authors create content online first, as Web pages, and lets readers turn individual pages or page collections into print documents. It uses wiki software as a lightweight and extensible content management system, so any page can be edited using any Web browser. It then uses LaTeX as the typesetting engine, thus providing print output of the highest quality. This paper examines the reasons for developing Wikipublisher, techniques and challenges faced in transforming Web content into print, and some wishes for the future. The project's home is `www.wikipublisher.org`.

1 Web-centred text

Wikipublisher was born of frustration over the gap between how I wanted to work and the capabilities of the tools I was using. In discussing this with like-minded colleagues, we came to the view that by re-thinking how we created and published information, we could be more productive and effective. We expressed this aspiration as a set of principles which underpin and inform every aspect of Wikipublisher's design:

Online first The World Wide Web enhances our ability to communicate, so most of our work ought to appear online first. Creating content online first makes it instantly and widely accessible, and encourages linking to other online resources. Yet most of our authoring tools are "print first" and turning print documents into HTML for publishing on the Web is hard to do well. Too many long documents are simply posted to the Web as PDF files. The links in an online document make it easy to navigate, yet print first authoring tools do little to encourage rich inter-document linking. So the first requirement is a system with support for the direct creation and editing of Web pages.

Print still matters If a Web page is worth reading, it is worth printing. The longer and richer the content, the more likely the reader is to print it. We may skim read a 50 page report online, but if we want to study it, we print it. If we want to deliver a printed and bound version, it needs to look good and be laid out for optimal readability. Yet few Web site designs appear to care what the printed form of the site looks like. Most appear to be designed assuming all the information on the site can be chunked into short, easily digested pieces. As readers, experience has taught us to have low expectations of the printed Web page.

One authoritative source The most up-to-date version is what appears on the Web page; the typeset PDF is a snapshot taken at a point in time. This means the printed page can never be newer than the Web page. This is in direct contrast to most publishing systems, where there are often 3 (and sometimes more) versions: the word processing or other source, a PDF snapshot of the word processing source, and a collection of Web pages generated (perhaps with edits) from the source. The more frequently the content changes and the more authors involved in creating it, the more important it becomes to have one source.

Wikipublisher is designed for people who write long, complex, richly linked documents, who wish to publish these in an accessible form on the Web and in print. The reader presses a "Typeset" button on the Web page and the system returns a PDF. By using LaTeX as the typesetting engine, Wikipublisher produces printed output of the highest quality.

2 We built it because ...

My company, Affinity Limited, is a small IT management consultancy. Like most professional services firms, our document production systems, based on Adobe FrameMaker, were geared to efficiently producing print documents such as letters, proposals, short papers, and reports. Our Web presence was essentially an online brochure — who we are, what we do, and how to contact us. Communication with clients primarily used e-mail to send document attachments back and forth. Several years ago, three things happened:

- we decided to make our work processes Web-centric wherever possible;
- open source wiki software became widely available, easy to install, and easy to use;

- Adobe decided not to offer a Mac OS X version of FrameMaker.

Apple's switch to Intel processors meant that OS 9 was dead and with it FrameMaker on Mac. Either we had to switch to Windows or find a replacement. We concluded that the least-worst alternative was to use LyX as a front-end to LaTeX, and while this would meet our print needs, it didn't really advance our aim to be more Web-centric. We had received very positive comments from clients when we introduced wiki software for some of our work. For example, when writing up interview notes, it is much easier and faster for everyone if we send a link to a Web page of the write-up, which the interviewee can edit using a Web browser. Wikis use simple textual markup to describe a page, which gets translated into HTML when the page is displayed. Each page carries an Edit link, which when clicked displays the content of the page as an editable Web form.

However, we have to deliver the interview notes as part of a printed and bound report, typically as an appendix. We experimented with "printable views" of the Web pages and quickly learnt that while the quality is good enough for an appendix, HTML plus CSS (cascading style sheets) produces printed output of a quality far lower than that produced by software such as FrameMaker or LaTeX. We also felt that it ought to be a lot easier to bundle up 10 interview Web pages into a single printed document. So we had succeeded in improving the doing part of the process, but at the cost of lowering the final output quality, and we still needed a solution for the main body of our reports. We concluded that if we were to make further advances in this direction, we needed a way to turn a Web page or page collection into a print document of at least the same quality as one expects from a modern word processor, with minimal manual intervention.

We decided to investigate the possibility of creating a report as a set of linked wiki Web pages and using some form of typesetting engine to repurpose this into a printable document, complete with cover page, table of contents, running headers and footers, and other print-oriented structures. With the help of a research and development grant from New Zealand's Foundation for Research, Science and Technology (FRST), we teamed up with the School of Mathematics, Statistics and Computing Sciences at Victoria University of Wellington to carry out the project. The grant allowed us to hire a graduate student on a part-time basis for just over a year, while he worked on a MSc. After a further round of enhancements and bug fixes, we can now produce letters, articles, reports, presentations and books, including images, tables, equations, citations and bibliographies — in other words, it has become a fully-fledged publishing system for Web and print documents. We have released the source code (a plug-in for the wiki engine and a separate typesetting server) and discovered that a surprising (to us) number of people find it useful.

3 TeXnical matters

Without TeX, the project would have been impractical. We considered using XSL-FO (formatting objects), but our not very scientific assessment found that at the time, none of the books on FO had been written using FO. We suspected that somewhere between the beautiful examples in the books and typesetting an arbitrary Web page, we would encounter impenetrable problems. TeX seemed to us a low risk and proven approach. Various books and PDF documents on LaTeX written with LaTeX gave us confidence that any problems we might find would have ready solutions.

3.1 Typesetting becomes a Web service

Wikipublisher works by re-purposing Web pages to a form of XML designed to describe printed material and then transforming the XML into LaTeX for typesetting. It builds on the following open source projects.

PmWiki We chose the PmWiki engine[1] because it is *markup agnostic* — that is, the administrator can control the input markup for structuring the content *and* the output markup the engine produces. The wiki continues to generate HTML for normal browsing, but when the reader requests a PDF document, we invoke a Wikipublisher plug-in which causes the wiki engine to generate Wikibook XML instead. The wiki's design means we can do this without any changes to the original source code.

tbook The tbook project[2] is an XML-based system designed to allow an author to create documents in XML and then transform these into a variety of different formats, including LaTeX. The original tbook system did not accommodate all the markup available in PmWiki, so we have extended the XML syntax quite significantly. We call the revised DTD (document type definition) "Wikibook" — any valid tbook XML document is valid Wikibook, but Wikibook supports constructs not found in tbook.

[1] `www.pmwiki.org`
[2] `sourceforge.net/projects/tbookdtd`

When a reader issues a request for a PDF, this goes to the Wikibook server, which asks the PmWiki server to generate the requested page(s) as Wikibook XML. The Wikibook server transforms the XML into LaTeX and thence into a PDF, which it returns to the reader. All an author or reader needs is a Web browser and PDF viewer; everything else happens on a Web server.

For sites requiring equations, we provide a plug-in for PmWiki that lets authors write TeX equations into a page, including automatically-generated equation numbers if required. On the Web, each equation is transformed into an image; in the PDF, the equations are just part of the document. We use the open source latexrender php library for this.[3]

3.2 Headings determine structure

Authors generally use heading markup to structure their pages. In HTML, there are no controls over how headings are used. Wikipublisher assumes that a page has sections, subsections and subsubsections; whatever 3 kinds of heading it finds on the page, it maps into this structure. In other words, it transforms the absolute HTML and wiki heading levels into relative section levels. This means different headings on different pages can be rendered the same way in print. PageA might start with `<h2>` while PageB starts with `<h3>`: both become sections. It also means the same heading might be rendered differently in different print contexts. If a page is being typeset as part of a list of pages, each page becomes a section, so the first heading on the page now becomes a subsection.

In other words, different authors can use different heading conventions (or the same author may use different conventions at different times), and let Wikipublisher sort it out and produce consistently laid out print documents. An author can create a long structured document using multi-level lists, like this (each item is a separate wiki page):

- Section A
  - Subsection A.1
  - Subsection A.2
- Section B
  - Subsection B.1
  - Subsection B.2

And so on. On a page like Subsection A.1, the first 3 heading levels now become subsubsection, paragraph and subparagraph. A list item can also be designated as an appendix, which means it and subsequent items are unnumbered.

[3] `www.mayer.dail.pipex.com/tex.htm`

3.3 Adapt the output on demand

Suppose that some of the readers want output on US letter paper while others need A4; some have duplex printers while others do not; some prefer indented paragraphs while others prefer space between paragraphs. Perhaps the author wishes to include a “draft” watermark. Wikipublisher provides a `<meta>` XML element with name/value attribute pairs to control these and other settings. An “options” button on the typesetting request form lets the reader control the look of the finished document, over-riding the default meta settings. A Canadian reader with a duplex printer and an Australian reader with a one-sided printer can each request the output he or she prefers.

The print metadata capability does the simple things like creating mirrored headers and footers on odd and even pages for duplex printing, or making all headings serif. It also does more sophisticated things; for example, if a reader requests A5 paper, it not only reduces the font size to compensate for the shorter line length, it also makes sure any images are shrunk, if they are too big to fit the smaller page. Indeed, whatever the page size, it makes sure the images fit the page.

While many of these print metadata settings are common to all document classes, such as choosing the fontset or watermark, some are class-specific. For example, when typesetting a book, the reader can choose the chapter heading style; when typesetting a letter, the reader can include or omit a return address; when typesetting an article, the reader can choose whether or not to number the sections.

3.4 Citations and bibliographies

Because the wiki content is “online first”, we cannot use any of the traditional bibliography tools like natbib. The problem we face is that LaTeX bibliography tools assume the author writes some variant of `\cite{key}` and LaTeX works out how to render the reference and sort the bibliography. On the other hand, to render a Web page with citations and a bibliography from wiki markup, the wiki engine has to solve these and other problems, then Wikipublisher has to tell LaTeX that we already know how to typeset the result. The wiki way to add a new entry to a bibliography is to cite it, just as the way to create a new page is to link to it:

- if the cited key exists, the wiki links to that entry in the page’s bibliography

- otherwise, it links to a “new citation” form — fill in the form and press Save

The Wikibook XML we generate contains everything LaTeX needs to know — the keys, the text of the link (numbered or author–year), the items (labelled) in the bibliography, sorted in the correct order. All LaTeX has to do is typeset the data. For example, the body text might contain:

```
<cite kind="sic" refid="Smith:2001">see
Smith (2001,&nbsp;p.6)</cite>
```

The corresponding bibliography entry text might be:

```
<item id="Smith:2001">Smith, A. 2001. ...
</item>
```

In the PDF, we want the `<cite>` to link to the `<item>` and we want the item to print with a hanging indent. To achieve this, we created a new LaTeX command:

```
\citesic{see Smith (2001~p.6)}{Smith:2001}
```

and a new kind of list environment for a preformatted, presorted bibliography, containing:

```
\abibitem{Smith:2001} Smith, A. ...
```

The `\citesic` command uses the `\hyperlink` command and `\abibitem` uses `\hypertarget` from the hyperref package. We use a similar approach for numbered references, except the `<item>` includes an attribute `style="n"` where `n` is the number of the reference.

Finally, Wikipublisher invites URL addresses to break on selected special characters, avoiding ugly white space in bibliographies (or indeed in regular text) containing references to Web sites. We achieve this by substituting `<discy kind="x" />` in the URL link text, where `x` is: hyphen (-), full stop (.), equals (=), underscore (_), or forward stroke (/); and generating:

```
\discretionary{}{x}{x}
```

This means the hyphenation character starts the line following the break point, thereby avoiding the ambiguity of an address line ending in a hyphen, which could cause a reader to wonder whether the hyphen is part of the address or was inserted during text hyphenation.

3.5 Tables

Wikipublisher currently provides support for three kinds of tables:

- simple tables float, their column widths are determined automatically, and text in the cells can be left justified, centred, or right justified; the caption, if there is one, is numbered
- long tables are like simple tables, except that they do not float and if the first row is headings (the wiki markup for the HTML `<th>` tag), this becomes a running header on the second and subsequent pages
- complex tables can contain any wiki markup the author deems it necessary to use (including simple tables); to handle these, we wrap the cell content in a LaTeX minipage environment, set all cell text to ragged right, and hope; if the author supplies percentage cell widths, we use these, otherwise we make the columns equal width

Authors are generally used to the fluid nature of wiki and HTML tables, and tend to assume that anything a Web browser can handle, the typesetting engine ought to handle too. By changing HTML attribute values, authors have fine control over table ruling, shading, colouring, and spacing on a Web page. There may also be site-wide table styles set in an external CSS file. Wikipublisher ignores almost all of these settings, and instead sets what we hope are reasonable and consistent defaults. In essence, it takes the *structure* of the table and ignores the *style*. This can be a strength, rather than a weakness — all the tables in a long document will look similar, whereas the original Web tables may show a great deal of stylistic variation.

We are investigating ways to handle rotated tables, especially long (multi-page) tables. The aspect ratio of modern computer screens is typically greater than 4:3 (the screen I am using at the moment is 1.6:1), so cell column widths which look fine through a Web browser tend to be too narrow on a portrait A4 page. Ideally, an author could assign a "wide" class to a table and Wikipublisher would print it rotated 90°, automatically splitting it across several pages if necessary, with column widths calculated from the available text height.

4 If we had a magic wand ...

And could make three wishes, what would we like to improve? Most of the time, "it just works"; once Wikipublisher is installed and configured, users can forget it is there until they need it, and when they need it, they get what they need, although this may not be precisely what they think they want.

4.1 Colour cite links correctly

I wish we could tell the hyperref package that the special `\citesic{link text}{key}` command we have defined is just a variation on the `\cite{key}` command, even though it invokes the `\hyperlink` command. At the moment, if you run Wikipublisher with the colorlinks option turned on, citation links come out in red (`linkcolor`) instead of green (`citecolor`). There is probably a simple way to modify the `wikibib.sty` file to check whether `colorlinks` is on and if so, to use `citecolor` for the colour of the generated hyperlink. Or preferably to define the `\citesic` command in such a way that hyperref recognises it as a citation link and colours it correctly. Any reader thinking "Oh, that's *easy*" might like to contact me. The `\citesic` command is defined as follows:

```
\newcommand\citesic[2]{\hyperlink{#2}{#1}}
```

4.2 Publish with style

I wish we had a general method for mapping the wiki's HTML style information into equivalent LaTeX print structures. Here are some of the things people do with wiki styles which at the moment do not work properly in Wikipublisher.

1. Define a style with a light green background, a dotted dark green border, text aligned right, with 0.5 em padding, applied to a paragraph.
2. Define styles for text decoration underline and line-through, even though there is a perfectly good structural markup to designate inserted, deleted, and highlighted text.
3. Set up "zebra tables" where alternate rows or columns are shaded. Combine alternate row and column stripes to create "hatched tables". Define cell ruling and padding.
4. Float text around a table using the table attributes `align="left"` or `align="right"`. Apply the `rowspan="n"` attribute to a table cell (although `colspan` works, `rowspan` does not).

An author can use wiki markup to define a named style consisting of any combination of HTML style attributes, and apply this style to any block, line or inline piece of text. Wikipublisher correctly handles styles like text size and colour, background colour, list item numbering and numbering style, and text alignment. There is a project awaiting us to analyse all the HTML style attributes, map these to their LaTeX equivalents (where these exist), and systematically define attribute transformations for any validly-styled text.

4.3 Typeset any Web page

I wish we could typeset pages from MediaWiki,[4] WordPress,[5] and other Web sites. In principle, if we can transform wiki markup, we ought to be able to transform HTML into Wikibook XML and then on into LaTeX. While this is superficially attractive, there are also some obvious problems. The HTML syntax makes no provision for semantic elements such as:

- figure captions
- footnotes
- marginal notes
- citations and references
- chapters, sections, subsections, appendices and other parts of a book

It also includes the widely used (and meaningless) `<div>` and `<span>` tags. In practice, one would have to rely on inferring structure from site-specific conventions for how class attributes are used. For example, if an image has `alt` text (as images should), use that as the caption. However, as more Web sites become database-driven, rather than using hand-crafted HTML, we can envisage developing different handlers (sets of rules) for different kinds of content management software, controlling the actions of a general-purpose HTML to Wikibook conversion engine. For example, suppose a university publishes an online journal in HTML or if *TUGboat* were to publish online first. A reader could choose articles of interest, from several online volumes, and request these as a single typeset PDF file. Potentially, high quality "print runs of one" become economic, perhaps with the ability to use a wider choice of LaTeX document classes than those currently supported.

We plan to seek some research and development funding to pursue this proposal. Unfortunately for us, FRST allocated almost all its 2007/08 grants budget in the first three months of the financial year and is only funding big companies at the moment. We have to wait until the 2008/09 financial year to apply.

5 Summing up

This article has described an open source software project which combines easy-to-use wiki software with the typesetting capabilities of LaTeX to create a simple, flexible, and powerful collaborative authoring and publishing system. Before the project started, I was an absolute LaTeX beginner; I have learnt how it works by looking at what the tbook XSL transformation did and watching how Donald Gordon, our part-time developer, enhanced this to handle the PmWiki markup set.

How does a small business justify investing this level of effort, when it would be easier to use Microsoft Word like almost everyone else? It is said that businesses change direction out of fear, greed, or boredom. It is true that we have won additional business as a result of the work, but not as much as we had hoped. It is also true that our clients see

[4] `www.mediawiki.org`
[5] `wordpress.org`

the project as innovative and pretty cool, so it has enhanced our reputation. As a research and development project it was a success, but commercialising the investment has been problematic. Currently, it is not generating revenue for us, but it has never been boring, it makes our working lives simpler, and it is a lot more fun than many of the things I get paid to do.

The project has taught us three important and in hindsight self-evident lessons.

Publishing online first alters the rules Like most other project-based consultancy practices, we write regular progress reports to our clients. These used to be paper letters; now they are wiki Web pages which can be typeset as letters. The shift from "documents" to "pages" creates a new perspective on the content. For example: they can contain links to related materials; the client can annotate them with comments; the entire history of the project is instantly accessible and searchable; and we can take print snapshots to store in a document management system or bind for physical distribution.

Open beats closed Using open source (free) software reduces the barriers to entry, allowing us to stand on the shoulders of giants. Open standards, especially the XML set of standards, enable interoperability between disparate systems like PmWiki and LaTeX. Open means that when we strike a problem, Google knows the solution: someone will have seen the problem before and left a trail on the Web for us to follow. The wiki markup and file format specifications are open and fully documented, so long term content curation and preservation are simply not a problem. In contrast, when we replace our last PowerPC-based Mac, a decade's worth of documents, stored in FrameMaker's proprietary format, will become inaccessible, unless we first convert them to a format like rtf.

Most people are not interested It is a minority of people who share our enthusiasm for what Wikipublisher can do. People who are familiar with HTML often do not see the point of transforming Web content into LaTeX to create a high quality print document. They see print as a disposable afterthought and consider that generating a printable page view using CSS is good enough. I find it surprising that people who pay careful attention to accessibility and readability principles for Web sites are happy to ignore these for printed material. On the other hand, most authors are comfortable with their word processor and see no reason to change their practice. If what they are writing will be published on a Web site, converting the document to HTML is someone else's problem. When their document is finished, they toss the dead sheep over the fence into the next paddock and forget about it.

The success of Microsoft Office in bringing typesetting to the mass market has had the unfortunate side-effect of entrenching typographic mediocrity in our culture. Most people are unaware of, and hence do not value, the correct use of ligatures, text justification algorithms, inter-paragraph separation, and the many other details which TeX looks after on our behalf. Competitors such as OpenOffice and Google Office are often judged by how well they replicate the layout of Word documents. So in a sense, Wikipublisher is over-engineered — it does a better job than it needs to.

If you are interested in Wikipublisher, you can try it online at `www.wikipublisher.org`, using any Web browser, or download the software and install it on any Unix or Linux server. I have PmWiki and the Wikibook PDF server running on my Apple MacBook. If you have any questions, you can contact me via the web site or the address below.

◇ John Rankin
Affinity Limited, Wellington, New Zealand
`john dot rankin (at) affinity dot co dot nz`