Making EPUB from WordPress (and other) web collections

Background

As part of Workpackage 3 I have been looking at WordPress as a way of creating scholarly monographs. This post carries on from the last couple, but it's not really about EPUB or about WordPress, it's about interoperability and how tools might work together in a Scholarly HTML mode so that people can package and repackage their resources much more reliably and flexibly than they can now.

While exploring WordPress I had a look at the JISC funded KnowledgeBlog project. The team there has released a plugin for WordPress to show a table of contents made up of all the posts in a particular category. It seemed that with a bit of enhancement this could be a useful component of a production workflow for book-like project, particularly for project reports and theses (where they are being written online in content management systems – maybe not so common now, but likely to become more common) and for course materials.

Recently I looked at Anthologize, a WordPress-based way of creating ebooks from HTML resources sourced from around the web (I noted a number of limitations which I am sure will be dealt with sooner or later). Anthologize is using a design pattern that I have seen a couple of times with EPUB, converting the multiple parts of a project to an XML format that already has some tools for rendering and using those tools to generate outputs like PDF or EPUB. Asciidoc does this using the DocBook tool-chain and Anthologize uses TEI tools. I will write more on this design pattern and its implications soon. There is another obvious approach; to leave things in HTML and build books from that, for example using Calibre which already has ways to build ebooks from HTML sources. This is an approach which could be added to Anthologize very easily, to complement the TEI approach.

So, I have put together a workflow using Calibre to build EPUBs straight from a blog.

Why would you want to do this? Two main reasons. Firstly, to read a report, thesis or course, or an entire blog on a mobile device. Secondly, to be able to deposit a snapshot of same into a repository.

In this post I will talk about some academic works:

- A thesis by Joss Winn who is on the JISCPress project.
- The Calibre ebook-convert tool and the Calibre server. I am running both of these as command line tools but there are versions you can run as desktop applications.
- Project work from my own website including blog posts from the jiscPUB project workpackage 3. I set up a page with two projects on it to show how these can be compiled into a book together.
- A set of open course materials on game design by Tony Hirst which I have imported into a test blog.

The key to this effort is the KnowledgeBlog table of contents plugin ktoc, with some enhancements I have added to make it easier to harvest web content into a book.

The results are available on a Calibre server I'm running in the Amazon cloud – just for the duration of this project. (The server is really intended for local use, the way I am running it behind an Apache reverse proxy it doesn't seem very happy – you may have to refresh a couple of times until it comes good). This is rough. It is certainly not production quality.

Title
Digital Worlds [Wed, 25 May 2011] EPUB (4.4 MB)
Projects @ ptsefton.com [Wed, 25 May 2011] EPUB (2.4 MB)
Ontogenesis [Thu, 19 May 2011] EPUB (0.6 MB)
The ptsefton.com Omnibus [Thu, 19 May 2011] EPUB (6.5 MB)
The Hand Painted Films of Margaret Tait by Joss Winn [Thu, 19 May 2011] EPUB (0.2 MB)

These books are created using calibre 'recipes': available here. You run them like this:

```
ebook-convert thesis-demo.recipe .epub --test
```

If you are just trying this out, to be kind to site owners `--test` will cause it to only fetch a couple of articles per feed.

I added them to the calibre server like this:

```
calibredb add --library-path=./books thesis-demo.epub
```

The projects page at my site has two TOCs for two different projects.

```
[ktoc cat="jiscPUB" title="Digital Monograph Technical Landscape study
#jiscPUB"  show_authors="false"  orderby="date" toc_author="Peter
Sefton"]

[ktoc cat="ScholarlyHTML" title="Scholarly HTML posts" orderby="date"
show_authors="false" toc_author="Peter Sefton" ]
```

I the title is used to create sections in the book, in both cases the post are displayed in date-order and I am not showing the name of the author on the page because that's not needed when it is all me.

The resulting book has a nested table of contents, seen here in Adobe Digital Editions.

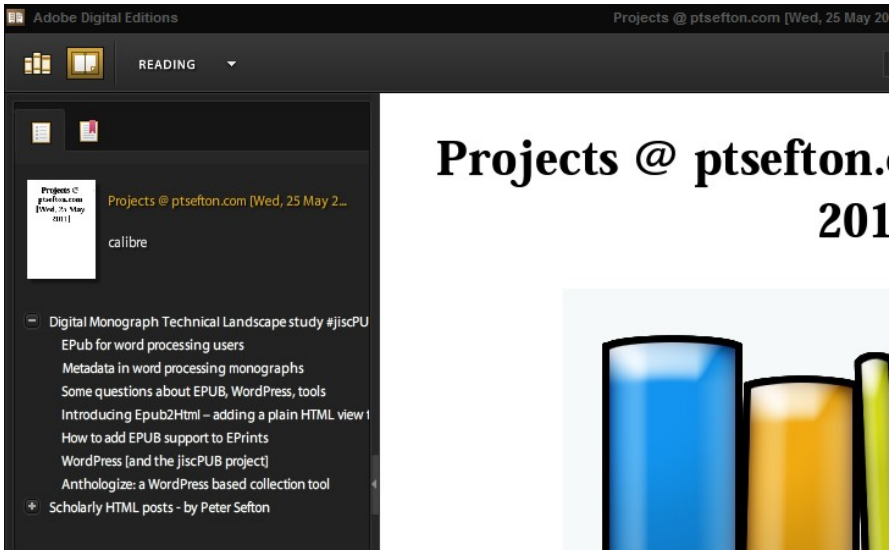

Illustration 1: A book built from a WordPress page with two table of contents blocks generated from WordPress categories.

Read on for more detail about the process of developing these things and some comments about the problems I encountered working with multiple conflicting WordPress plugins, etc.

The Scholarly HTML way to EPUB

The first thing I tried in this exploration was writing a recipe to make an EPUB book from a Knowledge Blog, for the Ontogenesis project. It is a kind of encyclopaedia of ontology development maintained in a WordPress site with multiple contributors. It worked well, for a demonstration, and did not take long to develop. The Ontogenesis recipe is available here and the resulting book is available on the Calibre server.

But there was a problem.

The second blog I wanted to try it on was my own, so I installed ktoc changed the URL in the recipe and ran it. Nothing. The problem is that Ontogenesis and my blog use different WordPress themes so the structure is different. Recipes have stuff like this in them to locate the parts of a page, such as `<p class='details_small'>`:

```
remove_tags_before = dict(name='p', attrs={'class':'details_small'})

remove_tags_after  = dict(name='div', attrs={'class':'post_content'})
```

That's for Ontogenesis, different rules are needed for other sites. You also need code to find the table of contents amongst all the links on a WordPress page, and deal with pages that might have two or more ktoc-generated tables for different sections of a journal, or parts of a project report.

Anyway, I wrote a different recipe for my site, but as I was doing so I was thinking about how to make this easier. What if:

The ktoc plugin output a little more information in its list of posts that made it easy to find no matter what WordPress theme was being used.

The actual *post* part of each page (ie not the navigation, or ads) identified itself as such.

The same technique could be extended to other websites in general.

There is already a standard way to do the most important part of this, listing a set of resources that make up an aggregated resource; the Object Reuse and Exchange specification, embedded in HTML using RDFa. ORE in RDFa. Simple.

Well no, it's not, unfortunately. ORE is complicated and has some very important but hard to grasp abstractions such the difference between an Aggregation, and a Resource Map. An Aggregation is a collection of resources which has a URI, while a resource map describes the relationship between the aggregation and the resources it aggregates. These things are supposed to have different URIs. Now, for a simple task like making a table of contents of WordPress posts machine-readable so you can throw together a book, these abstractions are not really helpful to developers or consumers. But what if there were a simple recipe/microformat – what we call a convention in Scholarly HTML – to follow, which was ORE compliant and that was also simple to implement at both the server and client end?

What I have been doing over the last couple of days, as I continue this EPUB exploration is try to use the ORE spec in a way that will be easy implement, say in the Digress.it TOC page, or in Anthologize, while still being ORE compliant. That discussion is ongoing, and will take place in the Google groups for Scholarly HTML and ORE. It is worth pursuing because if we can get it sorted out then with a few very simple additions to the HTML they spit out, *any* web system can get EPUB export quickly and cheaply by adhering to a narrowly defined profile of ORE – subject to the donor service being able to supply reasonable quality HTML. More sophisticated tools that do understand RDFa and ORE will be able to process arbitrary pages that use the Scholarly HTML convention, but developers can choose the simpler convention over a full implementation for some tasks.

The details may change, as I seek advice from experts, but basically, there are two parts to this.

Firstly there's adding ORE semantics to the ktoc (or any) table of contents. It used to be a plain-old unordered list, with list items in it:

```
<p><strong>Articles</strong></p>
<ul>
<li><a href="http://ontogenesis.knowledgeblog.org/49">Automatic
maintenance of multiple inheritance ontologies</a> by Mikel Egana
Aranguren</li>
<li><a href="http://ontogenesis.knowledgeblog.org/257">Characterising
Representation</a> by Sean Bechhofer and Robert Stevens</li>
<li><a href="http://ontogenesis.knowledgeblog.org/1001">Closing Down
the Open World: Covering Axioms and Closure Axioms</a> by Robert
Stevens</li>
</ul>
```

The list items now explicitly say what is being aggregated. The plain old <li> becomes:

```
<li  rel="http://www.openarchives.org/ore/terms/aggregates"
resource="http://ontogenesis.knowledgeblog.org/49">
```

(The fact that this is an `<li>` does not matter, it could be any element.)

And there is a separate URI for the Aggregation and resource map – courtesy of different IDs. And the resource map says that it describes the Aggregation – as per the ORE spec.

```
<div id="AggregationScholarlyHTML">

<div rel="http://www.openarchives.org/ore/terms/describes"
```

```
resource="#AggregationScholarlyHTML" id="ResourceMapScholarlyHTML"
about="#ResourceMapScholarlyHTML">
```

It is verbose, but nobody will have to type this stuff. What I have tried to do here (and it is a work in progress) is to simplify an existing standard which could be applied in any number of ways and boil it down to a simple convention that's easy to implement but that still honours the more complicated specifications in the background. (Experts this will realise that I have used an RDFa 1.1 approach here, meaning that current RDFa processors will not understand, this is so that we don't have to deal with namespaces and CURIES which complicate processing for non-native tools.)

Secondly the plugin wraps a element around the content for every post to label it as being scholarly HTML, this is a way of saying that this part of the whole page is the content that makes up the article, thesis chapter or similar. Without a marker like this finding the content is a real challenge where pages are loaded up with all sorts of navigation, decoration and advertisements, it is different on just about every site, and it can change at the whim of the blog owner if they change themes.

```
<div rel="http://scholarly-html.org/schtml">
```

Why not define an even simpler format?

It would be possible to come up with a simple microformat that had nice human readable class attributes or something to mark the parts of a TOC page. I didn't do that because then people will rightly point out that ORE exists and we would end up with a convention that covered a subset of the existing spec, making it harder for tool makers to cover both and less likely that services will interoperate.

So why not just use general ORE and RDFa?

There are several reasons:

Tool support is extremely limited for client and server side processing of full RDFa, for example in supporting the way namespaces are handled in RDFa using CURIES. (Sam Adams has pointed out that it would be a lot easier to debug my code if I did use CURIES and RDFa 1.0 – so I followed his advice, did some search and replacing and checked that the work I am doing here is indeed ORE compliant).

The ORE spec is suited only for experienced developers with a lot of patience for complexities like the difference between an aggregation and a resource map.

RDFa needs to apply to a whole page, with the correct document type – and that's not always possible to do when we're dealing with systems like WordPress. The convention approach means you can at least produce something that can become proper RDFa if put into the right context.

Why not use RSS/Atom feeds?

Another way to approach this would be to use a feed, in RSS or Atom format. WordPress has good support for feeds – there's one for just about everything. So you can look at all the posts on my website:

http://ptsefton.com/category/uncategorized/feed/atom

or use Tony Hirst's approach to fetch a singe post from the jiscPUB blog

http://jiscpub.blogs.edina.ac.uk/2011/05/23/a-view-from-academia-on-digital-humanities/feed/?

withoutcomments=1 .

The nice thing about this single post technique is that it gives you just the content in a content element – so there is no screen scraping involved. The problem is that the site has to be set up to provide full HTML versions of all posts in its feeds or you only get a summary. There's a problem with using feeds on categories too, I believe, in that there is an upper limit to how many posts a WordPress site will serve. The site admin can change that to a larger number but then that will affect subscribers to the general purpose feeds as well. They probably don't want to see three hundred posts in Google Reader when they sign up to a new blog.

Given that Atom (the best standardised and most modern feed format) is one of the official serialisation formats for ORE it is probably worth revisiting this question later if someone, such as JISC, decides to invest more in this kind of web-to-ebook-compiling application.

What next?

There are some obvious things that could be done to further this work:

- Set up a more complete and robust book server which builds and rebuilds books from particular sites and distributes them in some way, using Open Publication Distribution System (OPDS) or something like this thing that sends stuff to your Kindle.
- Write a 'recipe factory'. With a little more work the ScholarlyHTML recipe can be got to the point where the only required variable is a single page URL – everything else can be harvested from the page or over-ridden by the recipe.
- Combining the above to make a WordPress plugin that can create EPUBs from collections of in-built content (tricky because of the present calibre dependency but it could be re-coded in PHP).
- Add the same ScholarlyHTML convention for ORE to other web systems such as the Digress.it plugin and Anthologize. Anthologize is appealing because it allows you to order resources in 'projects' and nest them into 'parts' rather than being based on simple queries but at the moment it does not actually have a way to publish a project directly to the web.
- Explore the same technique in the next phase of WorkPackage 3 when I return to looking at word processing tools and examine how cloud replication services like DropBox might help people to manage book-like projects that consist of multiple parts.

Postscript: Lessons and things that need fixing or investiging

I encountered some issues. Some of these are mentioned above but I wanted to list them here as fodder for potential new projects.

- As with Anthologize, if you use the WordPress RSS importer to bring-in content it does not change the links between posts so they point to the new location. Likewise with importing a WordPress export file.
- The RSS importer applied to the thesis created hundreds of blank categories.
- I tried to add my ktoc plugin to a Digress.it site, but ran into problems. It uses PHP's simplexml parser which chokes on what I am convinced is perfectly valid XML in unpredictable ways. And the default Digress.it configuration expects posts to be formatted in a particular way – as a list of top-level paragraphs, rather than with nested divs. I will follow this up with the developers.
- Calibre does a pretty good job of taking HTML and making it into EPUBs but it does have its issues. I will work through these on the relevant forums as time permits.

- There are some encoding problems with the table of contents in some places. Might be an issue with my coding in the recipes.
- Unlike other Calibre workflows, such as creating books from raw HTML, `ebook-convert` adds navigation to each HTML page in the book created by a recipe. This navigation is redundant in an EPUB, but apparently it would require a source code change to get rid of it.
- It does something complicated to give each book its style information. There are some odd presentation glitches in the samples as a result of Calibre's algorithms. This requires more investigation.
- It doesn't find local links between parts of a book (ie links from one post to another which occur a lot in my work and in Tony's course), but I have coded around that in the Scholarly HTML recipes.

It will be up to Theo Andrew, the project manager if any of these next steps or issues get any attention during the rest of this project.

This post was written in OpenOffice.org, using templates and tools provided by the Integrated Content Environment project.