2010-11-16

Digital Publishing and Preservation Using XML

Welzenbach, Rebecca; Schaffner, Paul; Hawkins, Kevin

Exercise: TEI encoding 2

This exercise is based on one used as part of "From Text Encoding to Digital Publishing", a two-day workshop held at the National University of Ireland, Galway, and sponsored by the Digital Humanities Observatory, a project of the Royal Irish Academy.

The previous exercise gave an opportunity to encode a text whose structure is quite clear to the contemporary reader and to experiment with rendering this text in a web browser and in PDF format. In this exercise, we will work with an older text whose structure is less apparent at first glance: two facing pages of the beginning of a chapter of *Vegetable Dyes from North American Plants*.

The exercise assumes that you have a working installation of <oXygen/> XML Editor version 12 as well as the following files:

- `recipes.jpg`
- `recipes.win.txt`
- `recipes.mac.txt`
- `recipes-metadata.txt`

You might want to work with a printout of `recipes.jpg` rather than viewing it on your computer.

Part A: Getting started

1. Open the <oXygen/> (with a blue icon, not the "author" mode with a red icon).
2. Go to **File → New…**
3. In the hierarchy, click the **Framework templates** folder and then the **TEI P5** folder.
4. Choose **TEI Lite** to create a document using the latest version of TEI Lite with the minimum tags required to be valid and some boilerplate text filled in to guide you.
5. Go to **File → Save As…** to save the document. Save it as `recipes.xml` in the documents directory along with the other XML documents.

You might notice that the lines before the `TEI` element look different from how they appeared in the previous exercise.

After the XML declaration (whose presence is recommended as the first line of any XML file):

```
<?xml version="1.0" encoding="UTF-8"?>
```

you'll find a processing instruction for <oXygen/> that gives the location of the schema. It contains an absolute URL (to a version online) in this exercise, whereas in the previous exercise it contained a relative path to a file on your hard drive. Both ways are possible. In fact, <oXygen/> recognizes TEI documents and is able to validate even without an Internet connection using its own copy of the TEI Lite schema.

Let's add an "ID" to the root element:

6. Place your cursor after `xmlns="http://www.tei-c.org/ns/1.0"` but before the closing angle bracket (>).
7. Press the spacebar. A list of allowed attributes will appear.
8. Choose `xml:id` and give it the value `recipes`.

Check that the document validates:

9. Choose **Document → Validate → Validate**. There's also a button for this in the toolbar that looks like this:  . If your document is not valid, error messages will appear at the bottom of the <oXygen/> window, and the invalid sections of the document will be underlined with a red squiggly line.
10. To make the document easier to read, choose **Document → Source → Format and Indent**. There's also a button for this in the toolbar that looks like this:  .

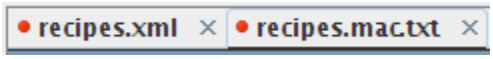
We'll start with the body of the source document (which will be represented within the `text` element) and do the metadata (which will be represented in the `teiHeader` element, above the `text` element) last.

Part B: Encoding the body of the document

In addition to the scanned page image of the text (`recipes.jpg`) to be encoded, we've also provided a plain-text transcription of the text so that you can copy and paste into your TEI document rather than typing everything yourself.

The file uses tabs and line breaks to approximately the layout of the text on the printed page, but as we learned in the previous exercise, this *whitespace* is ignored in XML. We will be using XML tags in order to make this structure explicit without relying on whitespace in the digital document to convey this structure. Likewise, plain text does not preserve any font effects (small caps or italics), but these can also be encoded using XML.

1. Open `recipes.win.txt` or `recipes.mac.txt` (depending on whether you're using Windows or Mac OS X) in <oXygen/>. This should now be your second document open within <oXygen/>; you can switch between the two files using the tabs:

   You might also choose to view the files side by side: choose **Window → Tile Editors Vertically**. To switch back, choose **Window → Stack Editors**.

The body of the TEI document has some fake content which we'll need to replace with the real content of our text:

2. Replace the "Some text here" paragraph and the `figure` element with opening and closing `div` tags, which indicate a structural division of text.

3. Add a `type` attribute with the value `chapter` to the `div` element. Remember that attributes and attribute values go on the opening tag, not the closing one:

```
<div type="chapter"></div>
```

The TEI allows you to use any value for the type attribute. A project should develop a *controlled vocabulary* of values to ensure consistency of encoding practice within and across XML documents.

Within this `div` that we've just inserted, we will insert the content of this chapter.

4. Within the `div` tags, insert a `head` element and put the name of the chapter between the opening and closing tags.
5. Following the `head`, insert a `p` element for the text of the first paragraph (and insert the text of the paragraph there).

The body of your TEI document should now look like this:

```
<body>
  <div type="chapter">
    <head>Recipes</head>
    <p>IN the following recipes, it is understood that
      the standard methods discussed in previous pages
      are the be used, unless some variation from this
      method is given. It should be noted that there
      may be differences in the mordanting as well as
      in the dyeing. The standard methods for these
      two processes are repeated in brief form here.</p>
  </div>
</body>
```

6. Now's a good time to save and validate.

After this paragraph are two recipes. Each has a heading of its own. Headings are a good indication that a new section of a text begins.

7. After the `p` element, insert two `div` elements—one for each recipe—each with `type="recipe"`. Don't copy and paste the plain text in quite yet.

The first recipe contains a table with four rows and two columns. Look at what follows this: it appears to be a paragraph with three sentences, but these sentences are actually steps in the recipe. In fact, if you look at the second recipe, you will see that it consists of only steps, which are not fully justified (with a smooth right edge) but instead are typeset so that no step is broken across lines. So while the typesetter of this book was inconsistent in the style used for the steps in the first and second recipes, we will encode both sets of steps using the same XML tags. That way, in our new digital edition of this work, we can ensure that steps are always displayed uniformly. (TEI has rich mechanisms for capturing the

imperfect appearance of the source document, but in this case we'll just capture the underlying structure of the text.

8. Insert a `head` for the title of the recipe.
9. Insert a `table` element by typing "<" and using the auto-complete feature. Once you do this, <oXygen/> automatically inserts a `row` with empty `cell` tags. (If you instead use the ⌘ + E (on a Mac) or Ctrl + E (in Windows) method, <oXygen/> won't fill in the child elements for you.) The schema requires that a `table` have at least one `row`, which in turn must have at least one `cell`, so <oXygen/> supplies all of this for you.
10. Fill out the first table—four rows, each with two cells.

The table should look like this:

```
<table>
  <row>
    <cell>Alum</cell>
    <cell>4 ounces</cell>
  </row>
  <row>
    <cell>Cream of tartar</cell>
    <cell>1 ounce</cell>
  </row>
  <row>
    <cell>Wool (dry weight)</cell>
    <cell>1 pound</cell>
  </row>
  <row>
    <cell>Water</cell>
    <cell>4 gallons</cell>
  </row>
</table>
```

11. Continue encoding the first recipe's list of steps.
12. Encode the list of three steps in the first recipe using the `list` element. Each step should be in an `item` element within the `list`.
13. Now would be a good time to use **Format and Indent** and then **Validate**. Note that if you run **Format and Indent** when there is a pair of opening and closing tags with no content (like `<div type="recipe"></div>`), it will collapse these tags into a single empty element: `<div type="recipe"/>`. If you want to insert text or markup within this element, remove the slash and <oXygen/> will uncollapse the tags.
14. Within the `div` for the second recipe:
    a. Insert a `head` for the title of the recipe
    b. Insert a `list` for the steps of the recipe.

At the top of page 33, we see a heading. While at first (especially without copies of other pages), this may look like a running header including the title or author, appearing at the top of each right-facing page. However, this is clearly a heading introducing a section on the alder shrub. So now we realize that everything in this chapter up to this point is itself a sort of introductory section, albeit without a heading such as "Introduction" labeling it explicitly. We should add a `div` to group all of this text together: it will

make it easier to see the hierarchy when encoding and could be useful if you are building a hierarchical table of contents using the div structure.

15. Wrap the `p` and the two instances of `<div type="recipe">` within a `<div type="intro">`.
16. Take a moment to Format and Indent and look at the structure of the document. Note that all of the `div` elements contain a `head` except one—the introduction that isn't labeled as being an introduction.
17. Following the closing tag of the `<div type="intro">`, insert a `<div type="plant">`. This new div will be a sibling, not child, of the intro div.
18. Inside the new `div`, insert a `head` for the text at the top of page 33.

What should we do with the italicized text in this heading? While TEI has an element (`hi`) for indicating that text is highlighted without saying why it's highlighted, it would be more in keeping with the spirit of XML to describe its structure. Why is this text italicized? The first word looks like a Latin name for the plant, so we can label this word as such. In fact, the first word—in all caps—is also a name. We can use the `name` element to tag text as containing a name, and we can use the `xml:lang` attribute to indicate the language of that name. But what is "Aune"? It's unclear, but it does appear to be a name.

Since in our edition of this classic work, we want to make sure users can search on any name of a plant that they might know. So we should carefully encode names.

19. Wrap "ALDER" in `<name xml:lang="en">`.
20. Wrap "Alnus" in `<name xml:lang="la">`.
21. Wrap "Aune" `<name>`.

Should the parentheses really remain outside of the name element? How should you encode a name in a possessive form (with an apostrophe and an s)? People's opinions on questions like this vary, and it really depends on what you want to do with the text afterwards. Since names in English don't change much (just possessive and sometimes plural forms), one advantage of leaving the parentheses (and possessive tagging) outside of the tags is that it's easy to write a program to pull all the names of out of the document without needing to worry about stripping this stuff from the name afterwards.

22. Continue encoding the text on page 33. Here are some elements that will be useful:
    a. `name` not just for the names of plants but als the names of places and peoples
    b. `cit` for a quotation from an outside work, which can contain:
        i. `quote` for the quotation itself
        ii. `bibl` for the bibliographic citation. Use <oXygen/> to find elements allowed as children of `bibl` which will be helpful in encoding the components of the citation.

    Note that the `cit` element may only contain child elements, not any text content. For that reason, you will need to include the dash before the bibliographic citation in either the `quote` or the `bibl` element.

    Be sure to read the recipe closely to distinguish prose from the list of steps!

We won't bother including the page number in the encoded text. Running headers, running footers, and page numbers are all called *forme work* and are rarely encoded because they can be automatically generated later. If you were encoding an important source document and wanted to represent the text as it is, even with errors, you might want to encode the forme work.

Part C: Filling out the header

A catalog record for this book is available at the URL given in `recipes-metadata.txt`. Open this webpage to get the information that you'll copy and paste into the `teiHeader`.

You'll recall from when we learned about the header earlier that there is a separate place in the header for describing the digital text and for the source from which this digital text was created:

1. <filedesc>: bibliographic info (*required*)
    1. <titleStmt> (*required*)
    2. <editionStmt> (*optional*)
    3. <extent> (*optional*)
    4. <publicationStmt> (*required*)
    5. <seriesStmt> (*optional*)
    6. <notesStmt> (*optional*)
    7. <sourceDesc> (*required*) ← description of the source

All other elements describe the TEI document itself.

2. <encodingDesc>: description of encoding practices (*optional*)
3. <profileDesc>: search terms (*optional*)
4. <revisionDesc>: record of changes (*optional*)

The `fileDesc` in <oXygen/>'s template contains only a `titleStmt` and a `sourceDesc` (since these are the only child elements required for `fileDesc`). Let's first fill out the `sourceDesc` using the information from the catalog record. The `sourceDesc` can contain either a free-text prose description in a `p` element or a more structured description using nested elements. For this exercise, we'll use `bibl` for a structured citation.

1. Replace the `p` and its boilerplate text with a `bibl`, containing the `author`, `title`, `pubPlace`, `publisher`, and `date` elements (which can occur in any order) with the appropriate text from the catalog record in each. Some notes about how to read a catalog record:
    a. Cataloging convention is to capital the initial word in a title and proper nouns, not most words in a title as in English. Feel free to preserve this practice or capitalize words to look more like a title typically does in English.
    b. The catalog record gives two dates: 1969 and 1943. 1969 is the date that this edition was published, whereas 1943 is the copyright date included on the title page verso. Just include 1969 within imprint.

Next we'll describe our digital edition of this work using the other child elements of `fileDesc`.

First we'll fill out the `titleDesc` with the following child elements:

2. Replace the content of the `title` element with an appropriate title for your digital edition—perhaps something like "A digital edition of an excerpt from *Vegetable dyes from North American plants*". So we will have a `title` within a `title`. You can do this in TEI!
3. Insert an `author` element for the author of the digital document. (The author of the digital document is the same as the author of the source document.)
4. Insert an `editor` element for the editor of the digital document (you!).

The `publicationStmt` element contains information about the publication of the digital text. Like the `sourceDesc`, allows either unstructured content in a `p` element or structured content using other elements.

5. Inside the `publicationStmt`, replace the content of the `p` element with structured elements to indicate that this digital text will be deposited in Deep Blue, the institutional repository of the University of Michigan, in 2011.

Congratulations! You've finished the exercise. If you still have time left and have also finished the previous exercise, look at the source document to see if there are other elements of the document you might want to encode and what utility this tagging that would provide for you.