Introductory TEI encoding 1

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.

In this exercise, you will:

- Get to know the <oXygen/> XML editor
- Encode an excerpt of an article, "The Act of Choice" by Richard Holton (Philosophers' Imprint 6.3), using the TEI Lite schema
- Transform your XML to XHTML (for display in a Web browser) and to PDF, using the basic TEI transformations that come bundled with the <oXygen/> software.

This exercise is a bit broader in focus than those that follow. You will encode a relatively large amount of text at a high level, using just a handful of elements, in order to get a sense of the structure of a TEI document and what you can do with it.

The exercise assumes that you have a working installation of <oXygen/> version 14 as well as a single directory (probably named encoding exercise 1) with the following files in it:

- holton_template.xml
- holton.pdf
- holton.mac.txt
- holton.win.txt
- teilite.rng

Part A: Getting started

1. Open the <oXygen/> editor (with a blue icon, not the "author" mode with a red icon).
2. If there are a lot of panels to the left and right of the main window content, let's first clear them out of the way: they're for advanced users. Close all of them. (If you ever want to reopen any of them, you can find them all under **Perspective → Show View**.)
   In <oXygen/>, open the file holton_template.xml. (**File →Open**)
3. To avoid overwriting this template (in case you want to consult it later), choose **File → Save As…** and save the file as myfirstTEI.xml in the same directory as the rest of the files. (It's important that this file be stored in the same location as the other files, so that <oxygen/> can find the schema against which to validate your document.)

You might notice an error message at the bottom of the <oXygen/> window about the when attribute. You can ignore this for now: we will fix this problem shortly.

Note that if you click the little triangles next to certain line numbers, you can "fold"

elements, hiding all of their content. If you fold line 5 and then line 41, you will see that, at the highest level, this TEI document consists of a `TEI` element (at "the root") and two child elements: `teiHeader` and `text`. In Part B we will fill in `teiHeader`, and in Part C we will fill in `text`.

Part B: Encoding the bibliographic information ("the header")

The `teiHeader` contains metadata about the TEI document and its source. It has four child elements: `fileDesc`, `encodingDesc`, `profileDesc`, and `revisionDesc` (we'll talk more about these later) According to the TEI Lite schema, which we're using in this exercise, only `fileDesc` is required, but if any others appear, they must be in the order just stated. For this exercise, we will create metadata for only the `fileDesc` using the following elements:

title	the title of the work (book, article, poem, etc.), including any subtitles
author	the author of the work
respStmt	statement of responsibility for the intellectual content of an edition (in this case, the electronic edition)
resp	phrase describing the nature of responsibility
name	proper noun or noun phrase
publicationStmt	information concerning the publication or distribution of an electronic or other text
sourceDesc	a bibliographic citation of the work being encoded

In your XML document, the skeleton of `fileDesc` is already in place. In this exercise you'll first fill in small pieces of information in `titleStmt` and `publicationStmt`, and then you'll encode most of `sourceDesc` yourself.

You should notice several elements containing two question marks, followed by a short phrase, and then another two question marks. This signals a place where you need to fill in a piece of information.

1. In between the opening and closing `title` tags, you'll find ??title here??. Replace this text with the title of the article, "The Act of Choice." Be sure that you leave the opening and closing tags in place. No need to enclose the title in quotation marks: this punctuation is not really part of the title but just a convention from print for displaying article titles. When we encode in XML, we are replacing the print convention (quotation marks) with <title> tags, which do the work of conveying that this is the title.

2. Do the same for `author` element: replace ??author here?? with the author's name, Richard Holton.

3. This document has three `respStmt` elements. One is already filled in, but you should supply the content of the name elements in the other two (with your name).

Your `titleStmt` should now look like this (but with your own name as the creator of the header and the encoder):

```
<titleStmt>
    <title>The Act of Choice</title>
    <author>Richard Holton</author>
    <respStmt>
        <resp>Creation of machine-readable text by</resp>
        <name>Rebecca Welzenbach</name>
    </respStmt>
    <respStmt>
        <resp>Header creation by</resp>
        <name>Mark U. Planguage</name>
    </respStmt>
    <respStmt>
        <resp>Encoded by</resp>
        <name>Mark U. Planguage</name>
    </respStmt>
</titleStmt>
```

Continuing …

4. Next, fill in in the `publicationStmt`. For today, we are all members of MPublishing, the unit of the library that publishes *Philosophers' Imprint*. All the information has already been filled in except today's date, which you should supply in any format you like as the content of the date element.

5. Dates can be written in many ways, and there is often good reason to allow flexibility in how you capture a date (for example, in order to be faithful how the date was written in an original manuscript). But a machine can't process this information (in order to sort a group of articles by date, for example), unless all the dates are encoded consistently. To make this possible, TEI provides the `when` attribute on the `date` element. This attribute is used to modify a `date` element by additionally giving the date in the standard format defined by the World Wide Web Consortium (W3C). The template contains YYYY-MM-DD as the default attribute value; change this to give today's date (using the Gregorian calendar with Arabic numerals).

6. The last element inside `fileDesc` is `sourceDesc`. This is the only element in `teiHeader` that describes the source (print, electronic, manuscript, etc.) of the text that is being encoded, rather than the XML document itself (e.g., this is the place to note the date that the article was published in a journal, rather than the date that you encoded this excerpt.) According to the TEI Guidelines, the element `bibl` contains "a loosely structured bibliographic citation." So, we can choose what information to encode, and how deeply to encode it. To complete this element, in <oXygen/>, open holton.mac.txt or holton.win.txt (depending on whether you're using a Windows or Mac machine.) You can move between this text file and myfirstTEI.xml using the tabs at the top of <oXygen/>'s editing window. This file contains a transcription of the article excerpt (which we'll use shortly), as well as some metadata. Copy the journal title, volume, issue

number, publication information, and date from holton.txt and paste it in between the opening and closing `bibl` elements in myfirstTEI.xml. This free text citation is a valid use of the `bibl` element, but we also can use more elements inside of `bibl` to be more explicit about the parts of our citation.

We'll start with the title:

7. Put the cursor in front of the title (before 'Philosophers' Imprint'). Type a left angle bracket (<). A menu will pop up showing the names of elements allowed at this point in the document according to the schema. (As you can see, there are quite a few!). We're going to use the title element, and as you start typing its name, the choice of elements matching your search will decrease. When there's only one element left, you can press **Enter** to choose it. <oXygen/> automatically inserts both the opening and closing tags. You'll need to move the closing tag to after the end of the title.

If you don't see a menu of elements available to you, it may mean that <oXygen/> can't find the schema that defines the rules for your XML document. Alert an instructor.

Let's use another method of inserting elements to tag the name of the volume and issue information, and the publication information. This method won't require us to move the closing tag.

8. Highlight 'Volume 6.'

9. Type ⌘ + E (on a Mac) or Ctrl + E (in Windows). A dialogue box opens with a list of the elements that may be inserted at this point according to the schema. Choose `biblScope`. The tags will be inserted before and after the text you highlighted.

10. Click inside the first `biblScope` tag and add a space. A menu will appear, showing all the attributes available to you; choose `type`**.** Then, click inside the quotation marks. A menu with the values available to you will appear; choose `vol`**.**
11. Do steps 8-10 for 'no. 3,' this time choosing `issue` instead of `vol` as the value of the type attribute of `biblScope`**.**
12. Finally, tag the publisher (<`publisher`>) publication place (<`pubPlace`>), and date (<`date`>) of publication, as you did in the <`fileDesc`>. Since we don't know the exact day that this article was published, we won't use the `when` attribute on `date` this time.

When you are marking up (encoding) texts, don't worry about extra spaces, tabs, or carriage returns (blank lines). These are all called 'whitespace', and XML ignores them, reducing all of these to a single space. Feel free to add blank lines or spaces between elements to make them more readable as you are working on them.

To help you make the document easy to read, choose **Document → Source → Format and Indent**. There's also a button for this in the toolbar that looks like this: 

13. Use **File → Save** to save changes made so far. Your `sourceDesc` should look something like this:

```
<sourceDesc>
    <bibl>
        <title>Philosophers' Imprint</title>
        <biblScope type="vol">Volume 6</biblScope>, <biblScope type="issue">no. 3</biblScope>
        <date>September
        2006</date>
        <publisher>MPublishing</publisher>
        <pubPlace>
            <address>
                <addrLine>University of Michigan Library</addrLine>
                <addrLine>Ann Arbor, MI</addrLine></address></pubPlace></bibl>
</sourceDesc>
```

Now, to check that you haven't made any errors in element names or how they are nested, we'll validate the document against the schema.

14. Choose **Document → Validate → Validate Document** or click the icon with the red check mark:  . In the status bar of the <oXygen/> window (at the bottom), it will say "Document is valid" or "Validation – failed." If the latter, <oXygen/> will tell you what errors exist and what line numbers to find these on. (The invalid sections of the document will also be underlined with a red squiggly line.) Fix any errors you encounter. (If you don't encounter any errors, try creating one by intentionally misspelling a tag. Once you validate, you'll see that <oXygen/> immediately detects this problem.)

Congratulations! You've just finished encoding your first header!

Part C: Encoding the body of the document

1. Check that you have removed all instances of '??' in your document in the course of encoding your header. Use **Find → Quick Find…** to do this. This menu command opens a small dialogue box at the bottom of the <oXygen/> window where you can type text to search for. As you see, the **Find** menu also offers more advanced search options. If you find any question marks, finish following the instructions in Part B.

2. Scroll down to the text element. (You may choose to collapse the teiHeader if you would prefer to keep this text out of sight while working with the body of the document. Use the little triangle to the left of the opening tag.

3. You should already have holton.txt at hand. Open holton.pdf, as well, as a reference for the structure of the article. The text in holton_template.xml currently has only one child element, `body`. (In TEI, a text can have a `front` and a `back` as well, used for front and back matter such as a preface, forward, appendix, or index.) Inside `body`, you'll find a `div` element, which indicates a division of text. Adding the `type` attribute to this `div` allows you to enter an arbitrary description of what kind of division this is. In this case, we'll use `article`:

```
<text>
    <body>
        <div type="article">
            <head>The Act of Choice</head>
```

This `div` contains a `head`, which contains the title of the article. Below it, the first section of the article has already been partially encoded for you. Each of the paragraphs is enclosed in a `p`, or paragraph, element.

4. Copy the rest of the text from the excerpt in holton.mac.txt or holton.win.txt, from "Choice, and how it differs from agency" to the end of the last paragraph before the notes. Paste all of this into myfirstTEI.xml, starting on a new line after the last `</p>`. By default, <oXygen/> will display each paragraph on a single line, and each line will probably run far out of your window. You may be tempted to press that "format and indent" button now, but try to resist: if you reformat these lines of text before wrapping them in their proper elements, <oXygen/> will collapse the lines together and you'll lose track of where one paragraph ends and where the next begins.
5. Look at holton.pdf to get a sense of the structure of the document. You should be able to see that the section that you just pasted into your XML document is a subsection of the article. Indicate this in the structure of your XML document by tagging it as a `div`. Insert a `type` attribute on this `div`, and give it a value that makes sense to you, such as `section`. (If you were encoding a full document, or many documents, you would use a controlled vocabulary to make sure that all other `divs` used to mark sections like this were assigned the same value of `type`. For our purposes, there is just one subsection to worry about, so you can call it whatever you want)
6. Using the methods for inserting XML elements given in Part B, tag the title of the section as a `head`
7. Now, insert `p` elements around each paragraph of text that you just pasted in. Each paragraph should start on its own line (and probably run far off to the right). By clicking three times, you can highlight an entire line at once, and enclose it in a `p` element. When you're done, you can use the **Format and Indent** feature to get your paragraphs aligned readably. You should have created 16 new paragraphs. Don't forget validate the document!
8. But wait! If you look at holton.pdf, you will see that one paragraph in this excerpt is really a blockquote! Find the paragraph beginning with "We asked people to tell us…." Instead of wrapping this paragraph in a `p` element, use the element `quote`. Then, on the opening `quote` tag, add a `type` attribute, and give it the value `block`.

Now we need to encode the notes associated with this excerpt.

9. The selection that we're working with includes the first 17 notes in the article. They're indicated throughout the text by numbers enclosed in parentheses. The content of these notes can be found at the end of the transcription of the article.
10. Find note 1 in holton.txt. Copy its contents and paste them into myfirstTEI.xml, replacing the (1), where the first reference to a note occurs.
11. Wrap the bit of text you just pasted text in a `note` element, and add two attributes to the opening tag.
    a. First, an `n` attribute. This indicates the note number, and the value of this attribute should be 1. Because the note number is captured by this attribute, you don't need to repeat it in the body of the note, so delete the"1." from the body of the note.
    b. Then, add a `place` attribute. This attribute describes where the note occurs in the source text. Choose the value bottom.

c. Repeat this with all 17 notes, increasing n by one each time. (If you get tired, bored, or run out of time, you can feel free to stop. Your document will validate, and the step of the exercise will work on the notes you have encoded, even if you don't do all of them)

12. Validate and save your changes.

Part D: Using Transformations to publish your TEI document

The Extensible Stylesheet Language (XSL) lets you transform an XML document into another type of XML document (using XSL Transformations or 'XSLT') or into a non-XML format (using XSL Formatting Objects or 'XSL-FO'). <oXygen/> includes some XSL-FO stylesheets for generating PDFs from TEI documents. Let's use one of the default stylesheets to make a PDF from myfirstTEI.xml.

1. In <oXygen/>, choose **Document → Transformation → Configure Transformation Scenario**.
2. Choose **TEI P5 PDF** and click **Transform now**. It will create myfirstTEI.pdf file in the same directory where myfirstTEI.xml is saved.
3. Open **myfirstTEI.pdf** in a PDF viewer.

Did it work? What do you see?

- Each note should be indicated in the text by its n value, supsercripted. The notes themselves appear at the bottom of the page as footnotes.
- The `<div type="section">` should be a subsection of `<div type="article">`

Now try the TEI P5 XHTML transformation scenario. This will convert your XML to XHTML that can be processed by a standard web browser.

What similarities and differences do you see between the PDF and the XHTML?

If you're interested and familiar with HTML, you may wish to open the transformed document, myfirstTEI.html in <oXygen/> (you will find it and your PDF in the same directory as your other XML documents), and look at how the TEI elements were mapped to HTML elements.

Part F: A challenge (in case you have time)

In holton.pdf, you'll notice many instances of italicized text that was lost in our encoding and transformation process. How would you encode these words to capture the meaning of the italics in your XML? (Hint: don't focus on recreating the appearance of italics, but instead on what the italics are supposed to convey to a human reader: emphasis? foreign word? something else?)