Exercise: 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 the first few pages of a journal article using the TEI Lite schema. TEI Lite is a simplified version of TEI that includes just some basic elements.
- Transform your XML to XHTML (for display in a web browser) and to PDF using the basic TEI transformations that come bundled with <oXygen/>.

The exercise assumes that you have a working installation of <oXygen/> XML Editor version 16.1 or later (an older version will work as well though may have different names for some functions) as well as the following files:

- `holton_pp_1-5.pdf`
- `holton_template.xml`
- `holton.win.txt`
- `holton.mac.txt`

You might want to work with a printout of `holton_pp_1-5.pdf` (which is five pages when printed on Letter size paper) 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. 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. (If you ever want to reopen any of them, you can find them all under **Window → Show View**.)
3. Use **File → Open…** to open `holton_template.xml`.
4. To avoid overwriting this template (in case you want to consult it later), choose **File → Save As…** and save a copy as `myfirstTEI.xml` **in the same directory of the original**. (For this exercise it's important that you save it in the same directory.)

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 error shortly.

In the main window, you'll see the text of `myfirstTEI.xml`. 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 the line 41, you will see that, at the highest level, this TEI document consists of a `TEI` element (at "the root" of the tree) 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. We'll learn more about this later in the day.

According to the TEI Lite schema, which we're using in this exercise, the only required child of `teiHeader` is `fileDesc`. It has a number of child elements: two are required (`titleStmt`, `publicationStmt`, and `sourceDesc`) and others are optional but must appear in a certain order in relation to the three required elements. They each have child elements, many of which can have even more descendants within them.

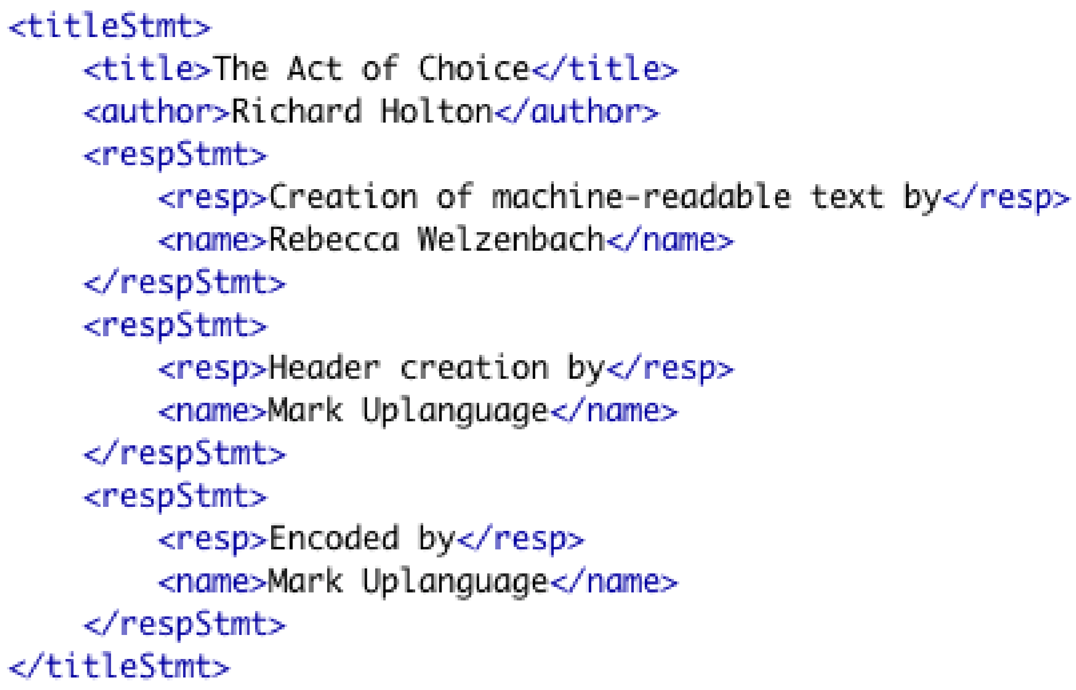
In your XML document, the skeleton of `fileDesc` is already in place. In this exercise you'll first fill in small pieces of information in `titleStmt` and `publicationStmt`, and then you'll encode most of `sourceDesc` yourself. As you go, we'll explain in brief what of these is used for.

You'll notice several elements containing two question marks followed by a short phrase and then two more question marks. We've used these to signal where you need to fill in a piece of information.

Here we go:

1. In between the opening and closing `title` tags (`<title>` and `</title>`), you'll find `??title here??`. Replace this with the title of the article: `The Act of Choice`. Be sure you leave the opening and closing tags in place. While a bibliographic citation for a journal article sometimes gives the title in quotation marks, it's better not to include those quotation marks here. In a sense, the opening and closing `title` tags function just like quotation marks, and if you wanted to add them when displaying your TEI document online or in print, you could do that with a stylesheet (which is separate from the TEI XML).
2. Do the same for the author element: replace `??author here??` with the author's name: `Richard Holton`. TEI has ways of separating first names (given names) from last names (surnames), but we won't bother for now.
3. This document contains three `respStmt` elements. One is already filled in, but you should supply your name as the content of the `name` elements inside the other two.

Your `titleStmt` should now look like this (but with your own name given as the creator of the header and as 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 Uplanguage</name>
    </respStmt>
    <respStmt>
        <resp>Encoded by</resp>
        <name>Mark Uplanguage</name>
    </respStmt>
</titleStmt>
```

Next we'll move on to the `publicationStmt`. This element is used to describe the publication of the TEI document itself, not the source document. Let's pretend that we work for Michigan Publishing, which will publish this TEI document.

The children of `publicationStmt` have all been filled out already except for one element: `date`. Since dates can be written many ways, the `when` attribute can be used to provide the date in a standard format, which allows a computer to list all dates in a TEI document, no matter the form they're given in, in chronological order. The TEI Lite schema can check that the dates are written in the standard format consisting of a four-digit year, a two-digit month, and a two-digit day of the month, which in the template is given as `YYYY-MM-DD`.

4. Change the value of the `when` attribute to give today's date in the form `YYYY-MM-DD`.
5. Fill in the content of the `date` element to write today's date in whatever human-readable form you please (for example, `January 16, 2015`).

The last element inside `fileDesc` in our template is `sourceDesc`. This describes the source of the encoded text rather than the TEI XML document itself. Let's pretend that this article previously appeared in print and that Michigan Publishing is reissuing it; therefore, we'll want to give a citation to the original print version here. The template already contains a `bibl` element, which is used for a bibliographic citation. But inside that element is just text without any additional tags labeling the components of the citation. Let's add them.

5. Put the cursor in front of the title (before `Philosophers' Imprint`). Type a single left angle bracket (<).

A menu will pop up showing the names of elements allowed at this point in the document according to the TEI Lite schema. (As you can see, there are quite a few!) We're going to use the title element.

6. With the pop-up menu still open, type the word "title". The menu will get shorter as it searches for elements matching what you type. When there's only once choice left, 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 of the journal.

Let's use another method of inserting tags for the volume and issue numbers and the date. This method won't require us to movee the closing tag:

7. Highlight `vol. 6`.
8. Type **⌘+ E** (on a Mac) or **Ctrl + E** (in Windows). A dialog box opens with a list of the elements that may be inserted at this point in the document according to the schema.
9. Choose `biblScope`. The tags will be inserted just before and after the text you highlighted.
10. Put the cursor between the `e` and the `>` of the opening `biblScope` tag (`<biblScope>`) you just inserted. Then press **Space**. A menu will appear showing all the attributes that can be added to the `biblScope` element.
11. Choose `type`. A new menu pops up showing the allowed values of the `type` attribute.
12. Choose `vol`.
13. Do steps 8 to 11 for the issue number, but instead of inserting the attribute value `vol`, choose `issue`.
14. Tag `Sept. 2006` with the `date` element. Use the when attribute like we did for the date of publication, only this time, the form to give will be `YYYY-MM` (without any day of the month).

When you are marking up (encoding) texts, don't worry about extra spaces, indentation, or blank lines. All of these things are called *whitespace* in XML, and XML software is supposed to ignore them in nearly all cases. Feel free to add blank lines or spaces between elements to make them easier to read as you work with them.

15. To make the whole document easier to read, choose **Document** ⟶ **Source** ⟶ **Format and Indent**. There's also a button for this in the toolbar that looks like this: .

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

16. Choose **Document → Validate → Validate**. There's also a button for this in the toolbar that looks like this: .

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 be underlined with a red squiggly line.

17. Fix any errors you encounter. If you don't encounter any errors, create one intentionally by changing the spelling of an element. Keep revalidating the document until you are able to fix the error.
18. Now is a good time to save your changes (**File → Save**).

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

19. So the header doesn't clutter up the <oXygen/> window while we continue working, fold the `teiHeader` element by clicking the little triangle next to it.

Part C: Encoding the body of the document

Following the `teiHeader` ("the header") is the `text` element, commonly called "the body". It is divided into an optional `front` element (for front matter like a preface or foreword), a required `body` element (for the body of the document proper), and an optional `back` element (for back matter like appendices or afterword).

The `front`, `body`, and `back` elements can all be divided into sections of text, for which the `div` element is most often used. (The template uses the special-purpose titlePage element as well, but we won't go into that today.) A `div` element generally begins with a `head`, which contains a heading for that section, but if no heading is present for that section (as is the case in our source document), you don't include a `head`.

1. Open `holton.win.txt` or `holton.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 above the main window with the text. You might also choose to view the files side by side: choose **Window** → **Tile Editors Vertically**. To switch back, choose **Window** → **Stack Editors**.
2. In the file you just opened, copy the rest of the text from "Choice, and how it differs from agency" to the end of the last paragraph before the notes.
3. Paste the text you copied into `myfirstTEI.xml`, starting on a new line after the `</div>`.

By default, <oXygen/> will display each paragraph on a single line, and each line will run far outside of your editing window. You may be tempted to press the **Format and Indent** button now, but don't: 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.

Look at the source document (`holton_pp_1-5.pdf`) to get a sense of the structure of the document. You will see that the section that you just pasted into your XML document is a section of the article of the article, which begins with a heading on page 2 and ends halfway down the left-hand side of page 5.

4. Indicate that the text you just pasted in is a division of the text by surrounding it with `div` tags.
5. Surround the heading for this section (which appears in bold in the source document) with `head` tags.

6. Click three times on each long line (each paragraph) of what you pasted in. This will select the whole line. Then use the **⌘+ E** (on a Mac) or **Ctrl + E** (in Windows) technique to tag each paragraph with the `p` element.
7. Use the **Format and Indent** feature to reformat the paragraphs to make them easier to read.
8. Validate the document to make sure you haven't made any tagging errors so far.

But wait! If you look at the source document, you will see that one paragraph in this section (beginning with "We asked people to tell us") is actually a blockquote rather than the author's words. Instead of using `p` for the blockquote, you can encode a blockquote in TEI using `<q rend="block">`.

9. Change the tags around the paragraph beginning "We asked people to tell us" to `<q>` and `</q>`.
10. Add a `rend` attribute on this `q` element, giving it the value `block`.

The section of the article that we pasted in includes the first 17 footnotes of the article. In `holton.win.txt` or `holton.mac.txt`, these are indicated by numbers inclosed in parentheses. The text of each footnote can be found at the end of `holton.win.txt` or `holton.mac.txt`.

11. Find note 1 at the end of `holton.win.txt` or `holton.mac.txt`. Copy the text and paste into `myfirstTEI.xml`, replacing the `(1)`, where the first reference to the note occurs. (Yes, we're putting the text of the footnote right in the middle of the text of the article.
12. Wrap the bit of text you just pasted in a `note` element.
13. Add the `n` attribute (for "number") to the opening `note` tag. This attribute indicates the note number, so in this case, give it the value `1`. Because the note number is captured by this attribute value, you don't need to repeat the note number anywhere in the element content (text that shows as black in <oXygen/>). So delete the "1" that appears just after the opening `<note>` tag.
14. Add the `place` attribute, with the value `bottom`, to `note` element as well. This attribute describes where the note occurs in the source text.
15. Re-validate the document to make sure you haven't made any errors.
16. On the other 16 footnotes, add the `note` element and the `n` and `place` attributes. While the value of `place` will be the same for all of them, be sure to increment the value of `n`. (If you're running short of time, feel free to skip the rest of the footnotes.)
17. Re-validate again 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 stylsheets for generating PDFs from TEI documents. Let's use one of the default stylesheets to make a PDF from `myfirstTEI.xml`.

1. In <oXygen/>, go to **Document → Transformation → Configure Transformation Scenario**.
2. Choose **TEI P5 PDF** and click **Transform Apply transformation**. The command may take a while to run, but <oXygen/> will eventually create a file called `myfirstTEI.pdf` in the same directory where `myfirstTEI.xml` is saved.
3. Open `myfirstTEI.pdf` in a PDF viewer.

4. Repeat step 1 for the **TEI P5 XHTML** transformation scenario, and open the resulting file in a web browser.

In each file, you will see that the note is indicated in the text by its `n` value displayed in superscript. The notes themselves appear at the bottom of the page as footnotes.

If you're familiar with HTML, you may wish to open the resulting XHTML file in <oXygen/> and look to see how the TEI elements were mapped to HTML elements.

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

In the source document, you'll notice italicized text that was lost in our encoding and transformation process. Since TEI, like many XML vocabularies, is more useful when describing content's structure rather than its appearance, TEI provides elements for marking up phrases that happen to be rendered in appearance based on the function they serve in the text, not just their appearance. Think about the italicized words you find. What do the italics convey to the reader? Emphasis? Foreign words? Something else?