Exercises : a little XPath, a little XSLT

2016-08-03

In these two exercises we will work on this short text, already marked up in XML. You will find it in the file Work/duBellay.xml — open it with Oxygen.

1 XPath

oXygen provides a simple way of exploring what XPath can do: at the top left of the screen just underneath the oXygen toolbar, you should see a little window labelled XPath 2.0. If you type an XPath expression into this and press the RETURN key, oXygen evaluates the expression you have typed returning any results in a new panel at the bottom of the screen. If you then click on one of these result lines, it will highlight the corresponding part of the document in the main window. Try it!

- Type `//title` into the box
- oXygen shows two lines, because there are two `<title>` elements in the document
- Click on the first one
- The title of the document is selected in the main editing window

Here's a little list of queries we will use XPath to solve :

1. what is the root element of this XML document?
2. what is the main document title?
3. who created this XML document ?
4. what original source was this document created from ?
5. when was the document last revised?
6. how many lines (`<l>`) are there in this document ?
7. how many archaic spellings (`<orig>`) does it contain ?
8. how many lines contain at least one archaic spelling?
9. in which lines does the word "jamais" appear ?

Feel free to try to solve these without looking at the answers below!

What is the root element ? XPath expression: `/*` [the slash alone selects the root node itself, the star shows the elements it contains]

Answer : TEI

What is the main document title? Xpath expression: `/TEI/teiHeader/fileDesc/titleStmt/title` or (amongst other possibilities) : `//titleStmt/title[1]`

Who created this XML document ? Xpath expression: `//titleStmt/respStmt/name`

Answer : in this case only one name is given; there might however be many.

A l'Ambicieux,

ET AVARE ENNEMY

DES BONNES LETTRES.

Sonnet.

Serf de Faueur, Eſclaue d'Auarice,
Tu n'heus iamais ſur toymeſmes pouuoir,
Et ie me veux d'vn tel Maitre pourueoir,
Que l'Eſprit libre en plaiſir ſe nouriſſe.
L'Air, la Fortune, & l'humaine Police
Ont en leurs Mains ton malheureux Auoir.
Le Iuge auare icy n'a rien à voir.
Ny les troys Seurs, ny du Tens la malice.
Regarde donc qui eſt plus ſoubaitable
L'ayſe, ou l'ennuy, le certain, ou l'inſtable.
Quand à l'honneur, i'eſpere eſtre immortel:
Car vn cler Nom ſoubz Mort iamais ne tumbe.
Le tien obſcur ne te promet rien tel.
Ainſi, tous deux ſerez ſoubz meſme Tumbe.

CAELO MVSA BEAT.

What is the source of this document ? Xpath expression: `//sourceDesc/*` or, to be more precise, `/TEI/fileDesc/sourceDesc/*`.

The * is needed if you want to see the elements which make up the <sourceDesc> (again, there might be many).

When was the document last revised? Xpath expression: `revisionDesc/change[1]/@when`. By convention, the first <change> should be the most recent. Its @when attribute gives a normalised version of the date.

Answer: 2011-05-23

How many lines (<l>) are there? Xpath expression: `count(//l)`. We use the XPath `count()` function to count things.

Answer: 10

How many archaic spellings (<orig>) ? Xpath expression: `count(//orig)`. Remove the `count()` to see them ...

Answer: 16

How many lines (<l>) have at least one archaic spelling (<orig>)? Xpath expression: `count(//l[orig])`

Answer: 2. Note that this counts only the lines which contain an <orig> directly. If you meant to include the lines where an <orig> appears within a <choice> as well, you should have said `count(//l[orig|choice/orig])`!

in which lines does the word "jamais" appear ? Xpath expression: `//l[contains(., 'jamais')]`

Answer: there are 2 such lines

2 XSLT

In this second exercise, we'll explore XSLT using oXygen.

2.1 Transformation into HTML

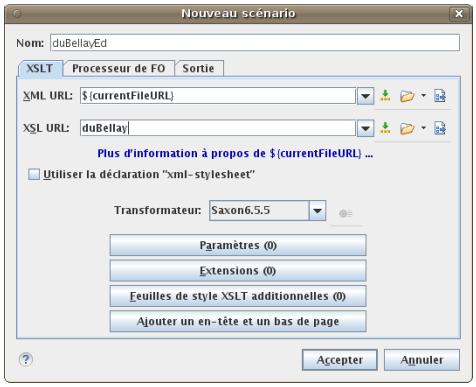
A traditional use for XSLT is to transform a TEI XML document into HTML for display in a web browser.

The simplest way of using XSLT in oXygen is to set up what is called a transformation scenario, which is an association between an XML document, an XSLT stylesheet, and various parameters. Proceed as follows: i

1. Open the file duBellay.xml in oXygen.

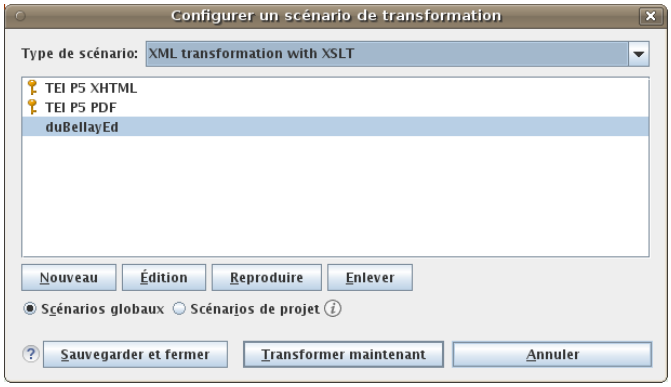
2. On the Document menu, select Transformation, and then Configure transformation scenario(s). Or type `ctrl-shift-C`. Or click the spanner button to the right of the red triangle on the toolbar.

Click the New button, and select XML transformation with XSLT to open the New Scenario dialog.



- Type duBellay.xsl in the XSL URL box.
- Select one of the xslt-2 processors e.g. Saxon-HE 9.6.07 from the scrolling menu labelled Transformer
- Click the Output tab
- Type output.html in the Save As box
- Tick the Open in Browser checkbox
- Click the Accept button

You've now created a Transformation Scenario called duBellay.



- Click the Transform now button
- At the bottom of the screen, the message Transformation successful should appear ...
- .. and after a brief delay, your web browser will open the file output.html which you just created
- Smart, isn't it ? Well, maybe there's still a but of work to be done...

2.2 An XSLT stylesheet

An XSLT stylesheet is an XML document like any other. So it can be edited in Oxygen.

- Open the file duBellay.xsl in oXygen and look at the templates it defines.
- So far, only one template is defined. It will match the element <TEI>, and output the tags <html> and </html>, with in between the result of an <xsl:apply-templates>

- ... which will check all templates available in the stylesheet, Since there aren't any, it will default to outputting just the text fragments in the document.
- ... and if you return to your browser and ask it to display the source of the HTML file, you'll see that is exactly what has been done.

To work ! We need more templates. First, let's extract the title of the document from the TEI Header and use it to title the HTML file as well.

In HTML, the title of a document is tagged `<title>` and appears within a <head> element, so we must add these. Edit the XSLT file as follows:

- After `<html>`, type <head>
- oXygen helpfully adds a closing tag. Good. Add a `<title>` within the <head> element.
- To find the title of the document, as you know, we will navigate from the document root (TEI) to a `<title>` element in a `<titleStmt>` in the `<fileDesc>` in the `<teiHeader>`. We've already seen an XPath to do that.
- Type `<xsl:value-of select="teiHeader/fileDesc/titleStmt/title[@type='main']"/` inside the `<title>` element in your stylesheet.
- Now we will change the `<xsl:apply-templates>` element so that it only looks at the `<text>` by adding `select="text"`. Enclose it with an HTML <body> element (use CTRL-E!)
- Click the Format and Indent button to tidy up. Your stylesheet should now look (more or less) like this :

```
1  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
2    xpath-default-namespace="http://www.tei-c.org/ns/1.0" version="2.0">
3
4    <xsl:template match="TEI">
5      <html>
6        <head>
7          <title>
8            <xsl:value-of select="teiHeader/fileDesc/titleStmt/title"/>
9          </title>
10       </head>
11       <body>
12         <xsl:apply-templates select="text"/>
13       </body>
14     </html>
15   </xsl:template>
16
17 </xsl:stylesheet>
18
```

- Click the diskette icon, or type CTRL-S, to save the changes you just made
- Click the duBellay.xml tab to return to your XML document .
- Click the red triangle icon to run your transformation scenario again (or type CTRL-SHIFT-T)

- Progress ! There is a title, at the top of your browser window, and the display shows only the text of the document, with no TEI header information.

We just need a few more templates.

- Return to your stylesheet window (click the duBellay.xsl tab).
- Add
  - one template for <l>, which will add an HTML
 tag after each line
  - one template for <head>, which will transform the XML <head> into an HTML <h2>
  - one template for <lg>, which will surround them with <p>, and also output the number of the stanza
- Here are the templates you need ... do you understand how each one works?

```
17  <xsl:template match="l">
18    <xsl:apply-templates/><br/>
19  </xsl:template>
20
21  <xsl:template match="head">
22    <h2><xsl:apply-templates/></h2>
23  </xsl:template>
24
25  <xsl:template match="lg">
26    <xsl:number/>
27    <p><xsl:apply-templates/></p>
28  </xsl:template>
29
```

2.3 Handling <choice> elements

XSLT will also help us process the <choice> elements in our document more intelligently. We could for example simply suppress any <orig> which appears inside a <choice> (or any <reg> if you prefer).

All we need is to add a template like this : <xsl:template match="choice/orig"/>. Try it. Do you understand how it works? If so, you are well on your way to becoming an XSLT Xpert!