


<oo> → <dh> Digital humanities

Maintained by: David J. Birnbaum (djbitt@gmail.com) 

Last modified: 2016-10-20T21:24:33+0000

XSLT assignment #4

The input text

For this assignment you will be working with Shakespearean sonnets, which you can download from <http://dh.obdurodon.org/shakespeare-sonnets.xml>. You should right-click on this link, download the file, and open it in <oXygen/>.

Using modal XSLT

What happens if you need to process the same nodes in your document in two different ways? For example, what happens if you need to output them as list items in a table of contents, but also as headers or text in the main body of your document, below the table of contents? Wouldn't it be handy to be able to have two completely different template rules that match exactly the same elements, one rule to output the data as list items in the table of contents and the other to output the same data as headers? You can write two template rules that will match the same nodes (have the same value for their **@match** attribute), but how do you make sure that the correct one is handling the data in the correct place?

For this assignment we would like you to get some experience working with modal XSLT. As is explained at <http://dh.obdurodon.org/modal-xslt.html>, modal XSLT allows you to output the same parts of the input XML document in multiple locations and treat them differently each time. That is, it lets you have two different template rules for processing the same elements or other nodes in different ways, and you use the **@mode** attribute to control how the elements are processed *at a particular place* in the transformation.

Overview of the assignment

For this assignment you want to produce an HTML version of the sonnets with a table of contents at the top. The table of contents should have one entry for each sonnet, which gives the number of the sonnet and the first line. Below the full table of contents (one line for each sonnet) you should render the complete text of all of the sonnets. You can see our output at <http://dh.obdurodon.org/shakespeare-sonnets.xhtml>.

How to begin

Begin by forgetting about the table of contents, and concentrate on just outputting the full text of the sonnets. This is just like the XML-to-HTML transformations you have already written, and you'll use regular template rules (without a **@mode** attribute) to perform the transformation. In our HTML output (scroll down past the table of contents, to where the full text of the sonnets is rendered), the roman numeral before each sonnet is an HTML **<h2>** element and the body of each sonnet is an HTML **<p>**

element. To make each line of the poems start on a new line, we add an HTML empty `
` (“[line] break”) element at the end of each line within the stanza. If you don’t include the `
` elements, the lines will all wrap together in the browser. Here’s the HTML output for one of our sonnets:

```
<h2>VI</h2>
<p>Then let not winter's ragged hand deface,<br/>
    In thee thy summer, ere thou be distill'd:<br/>
    With beauty's treasure ere it be self-kill'd.<br/>
    Make sweet some vial; treasure thou some place<br/>
    That use is not forbidden usury,<br/>
    Which happies those that pay the willing loan;<br/>
    That's for thy self to breed another thee,<br/>
    Or ten times happier, be it ten for one;<br/>
    Ten times thy self were happier than thou art,<br/>
    If ten of thine ten times refigur'd thee:<br/>
    Then what could death do if thou shouldst depart,<br/>
    Leaving thee living in posterity?<br/>
    Be not self-will'd, for thou art much too fair<br/>
    To be death's conquest and make worms thine heir.
</p>
```

The fine print: Don’t worry if your HTML output isn’t wrapped the same way ours is, if it puts the empty line break elements at the beginnings of lines instead of at the ends, or if it serializes (spells out) those empty line break elements as `
</br>` instead of as `
`. Those differences are not *informational* in an XML context. You can open your HTML output in `<oxyen>` and pretty-print it if you’d like, which may make it easier to read, but as long as what you’re producing is valid HTML and renders the text appropriately, you don’t have to worry about non-informational differences between your markup and ours.

More fine print: You need a line break only between lines, which is to say that you don’t need a `
` element at the end of the last line of the sonnet because that’s the end of the containing `<p>`, and not between lines. In our solution we used an `<xsl:if>` element to check the position of the line and output the `
` only for non-final lines. If you’re feeling ambitious, you can look up `<xsl:if>` at http://www.w3schools.com/xml/xsl_if.asp or in Michael Kay and perform this check yourself. If not, you can just output the `
` element after all lines of the sonnet, including the last. That’s not really considered good HTML style, and you don’t want to do it in your own projects, but it won’t interfere with the legibility in the browser and we’ll let it pass for homework purposes.

Once your sonnets are all being formatted correctly in HTML, you can add the functionality to create the table of contents at the top.

Adding the table of contents

The template rule for the document node in our solution, revised to output a table of contents before the text of the sonnets, looks like the following:

```
<xsl:template match="/">
  <html>
    <head>
      <title>Shakespearean sonnets</title>
    </head>
    <body>
      <h1>Shakespearean sonnets</h1>
      <h2>Contents</h2>
      <ul>
        <xsl:apply-templates select="//sonnet" mode="toc"/>
      </ul>
      <hr/>
    </body>
  </html>
</template>
```

```
        <xsl:apply-templates/>
    </body>
</html>
</xsl:template>
```

The highlighted code is what we added to include a table of contents, and the important line is **<xsl:apply-templates select="//sonnet" mode="toc"/>**. This is going to apply templates to each sonnet *with the @mode attribute value set to "toc"*. The value of the **@mode** attribute is up to you (we used "toc" for "table of contents"), but whatever you call it, setting the **@mode** to any value means that only template rules that also specify a **@mode** with that same value will fire in response to this **<xsl:apply-templates>** element. Now we have to go write those template rules!

What this means is that when you process the **<sonnet>** elements to output the full poems, you use **<xsl:apply-templates>** and **<xsl:template>** elements without any **@mode** attribute. To create the table of contents, though, you can have **<xsl:apply-templates>** and **<xsl:template>** elements that select or match the same elements, but that specify a mode and apply completely different rules. A template rule for **<sonnet>** elements in table-of-contents mode will start with **<xsl:template match="sonnet" mode="toc">**, and you need to tell it to create an **** element that contains a roman numeral and a first line, both fetched from the sonnet in the input XML file. The rule for those same elements not in any mode will start with **<xsl:template match="sonnet">** (without the **@mode** attribute). That rule will create the **<h2>** header to hold the roman numeral and then output the full text of the poem in a **<p>**, with **
** elements between the lines. In this way, you can have two sets of rules for sonnets, one for the table of contents and one for the body, and use modes to ensure that each is used only in the correct place.

*Remember: both the **<xsl:apply-templates>**, which tells the system to process certain nodes, and the **<xsl:template>** that responds to that call and does the processing must agree on their mode values. For the main output of the full text of every poem, neither the **<xsl:apply-templates>** nor the **<xsl:template>** elements specifies a mode. To output the table of contents, both specify the same mode.*