


<oo> → <dh> Digital humanities

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

Last modified: 2017-03-01T03:29:03+0000

XSLT assignment #5

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/>. You will be building on our [XSLT assignment #4](#), and you can take your stylesheet from that assignment and modify it for this one.

Overview of the assignment

For your last assignment you used the XSLT **@mode** attribute to create a table of contents for the Shakespearean sonnets, using the first line of each sonnet as a surrogate for the title (since they don't have real titles). Our output is at <http://dh.obdurodon.org/shakespeare-sonnets.xhtml>.

What's a table of contents good for anyway

In a digital edition, we can just do a full-text search and scroll in the browser, so we don't really need a table of contents at all. We can search for a roman numeral, we can search for the text of the first line of a sonnet, or we can search for a memorable phrase. But suppose we want to produce a paper edition, where the only organized access our users will get is the organization we decide to give them. What would be a useful table of contents or index?

A table of contents in the same order as the full text (numerical order), which is what we produced in the last assignment, duplicates the ordering information in the plain text. How useful is that? If we want to find a sonnet with a low number, we already know without a table of contents that we should look near the beginning. On the other hand, it's very common in published poetry collections to include an index of first lines, sorted in alphabetical order, so that a user who remembers just the first line of a poem can find it easily.

For this assignment we're going to enhance our output from the last assignment in the following ways:

- We're going to create links between the items in the table of contents and the sonnets, so that you can click on a line and be taken immediately to the corresponding sonnet.
- We're going to alphabetize our list of first lines, so that the table of contents will be sorted alphabetically, instead of in numerical order.
- As long as we were sorting the lines by order of appearance in the collection, that is, by numerical order of sonnet, it made sense to put the sonnet number (the roman numeral) first. For example, the sixth entry in our original list read "VI. Then let not winter's ragged hand deface,". If we're now going to sort by the first line of text, though, having those roman numerals at the far left edge of the entry will be disorienting, since they'll obscure the fact that we're using alphabetical order. We do want to retain the roman numerals (after all, the sonnet numbers are meaningful for Shakespeare scholars), but we're

going to move them to the *end* of the line, so that when the user reads down the left edge of the table of contents, scanning for a particular first line, the alphabetic order will be immediately accessible.

Our HTML output is at <http://dh.obdurodon.org/shakespeare-sonnets-sorted.xhtml>.

The tools we need

To create links between the first lines in the table of contents and the sonnets in the full text section of the page below we're going to use *attribute value templates* (AVT). If you haven't done so already, you should read about AVTs at <http://dh.obdurodon.org/avt.xhtml>.

To sort the table of contents we're going to use `<xsl:sort>`.

When we sort the first lines, they won't sort correctly for a quirky reason. We're going to fix that using the XPath `translate()` function, which we discuss below.

How HTML linking works

The `` items in the table of contents should include `<a>` ("anchor") elements, which is how HTML identifies a clickable link. An anchor that is a clickable link has an `@href` attribute, which points to the target to which you want to move when you click on the link. For example, the table of contents might contain the following list item for Sonnet VI:

```
<li><a href="#sonnetVI">Then let not winter's ragged hand deface, (VI)</a></li>
```

HTML `<a>` elements that have `@href` attributes normally appear blue and underlined in the browser, to advertise that they are links. The *target* of a link can be any element that has an `@id` attribute that identifies it uniquely. If you click on this line in the browser, the window will scroll to the element elsewhere in the document that has an `@id` attribute with the value "sonnetVI". In our case, we've assigned that `@id` attribute value to the `<h2>` for that sonnet in the main body:

```
<h2 id="sonnetVI">VI</h2>
```

Adding links to your output

You should first read our page on [Attribute value templates \(AVT\)](#), which describes a strategy you can use to create a unique `@id` attribute for each sonnet. For this task we gave our sonnets `@id` values that were a concatenation of the string "sonnet" and the roman numeral of the sonnet, e.g., "sonnetVI" for Sonnet #6. We attached those `@id` attributes to the `<h2>` elements that we used as titles for each sonnet in the body of our page, e.g., `<h2 id="sonnetVI">`. Meanwhile, in the table of contents at the top we created `<a>` elements with `@href` attributes that point to these `@id` values. *The value of the `@href` attribute must begin with a leading "#" character, but that "#" must not be part of the value of the `@id` attribute to which it points.* For example,

```
<li><a href="#sonnetVI">Then let not winter's ragged hand deface, (VI)</a></li>
```

means if the user clicks on this line, the browser will scroll to the line that reads `<h2 id="sonnetVI">` in the main body of the page. *Remember: the value of the `@href` attribute begins with "#", but the value of the*

corresponding **@id** attribute on the **<h2>** element you want to scroll to doesn't.

Armed with that information, you can take your answer to the main assignment and, using AVTs, modify it to create the **<a>** elements with the **@href** attributes and the **@id** attributes for the targets.

Sorting

An index of first lines in a collection of poems is usually alphabetized, because that's how humans look things up in that kind of list. To learn how to sort your table of contents before you output it, start by looking up **<xsl:sort>** at https://www.w3schools.com/xml/xsl_sort.asp or in Michael Kay. So far, if we've wanted to output, say, our table of contents in the order in which they occur in the document, we've used a self-closing empty element to select them with something like **<xsl:apply-templates select="//sonnet"/>**. We've also said, though, that the self-closing empty element tag is informationally identical to writing the start and end tags separately with nothing between them, that is, **<xsl:apply-templates select="//sonnet"></xsl:apply-templates>**. To cause the elements being processed to be sorted first, you need to use this alternative notation, with separate start and end tags, because you need to put the **<xsl:sort>** element between the start and end tags. If you use the first notation, the one with a single self-closing tag, there's no "between" in which to put the **<xsl:sort>** element. In other words, you want something like:

```
<xsl:apply-templates select="//sonnet">
  <xsl:sort/>
</xsl:apply-templates/>
```

As written, the preceding will sort the **<sonnet>** elements alphabetically by their text value. As you'll see at the sites mentioned above, though, it's also possible to use the **@select** attribute on **<xsl:sort>** to sort a set of items by properties other than alphabetic order of their textual content.

After the sort

At this point we'd make other adjustments in the output. The original table of contents begins with a roman numeral, but if you're going to sort the table of contents, you want the text of the first line of the poem at the left side of the line, not preceded by the roman numeral, so that you can see the alphabetic order easily. Putting the roman numeral first would make it harder to discern the alphabetization, since the user wouldn't be able to see it by just glancing down the left margin. For that reason, you should now adjust the output to put the roman numeral after the text of the line, in parentheses.

Using **translate()** to fix the sort order

If you sort the first lines alphabetically according to their textual value, there will be one error. The first line of Sonnet #121, "Tis better to be vile than vile esteem'd," will show up first because in the internal representation of characters in the computer, the single straight apostrophe is "alphabetically" earlier than all of the letters. We can fix this by using **translate()** to strip the apostrophe for sorting purposes, but not for rendering. That is, we can sort as if there were no apostrophe, while still printing the apostrophe when we render the line.

We can't easily translate away an apostrophe, though, because quotation marks have special meaning in XPath. For the purpose of this assignment, you can ignore this one missorted line. If you're feeling ambitious, though, read Michael Kay's answer at <http://p2p.wrox.com/xslt/50152-how-do-you-translate-apostrophe.html> and see whether you can apply it to fixing this problem.

Finishing touches

Some lists of first lines of poetry put quotation marks around the lines. We haven't done that in our solution, but if you'd like to add it, you should use the HTML `<q>` ("quoted text") element, instead of outputting the raw quotation marks as plain text.

Oh, and did we mention CSS? Can you attach a CSS stylesheet to your output to make it look more interesting than what you get by default in a web browser?