<oo>→<dh> Digital humanities

Maintained by: David J. Birnbaum (djbpitt@gmail.com)
Last modified: 2017-04-03T15:31:32+0000

XQuery assignment #2

About FLWOR constructions

This assignment requires the use of a FLWOR construction, so you might want to review the assigned reading before tackling it. Be sure that you understand what each of the components of a FLWOR does, and how they interact. For example, you can get the title of every play in the corpus with a single XPath statement:

```
xquery version "3.0";
declare namespace tei="http://www.tei-c.org/ns/1.0";
collection('/db/apps/shakespeare/data')//tei:titleStmt//tei:title
```

To get the same result with a FLWOR expression (alphabetizing the titles on the way), you might use:

```
xquery version "3.0";
declare namespace tei="http://www.tei-c.org/ns/1.0";
let $plays := collection('/db/apps/shakespeare/data')
for $play in $plays
order by $play
return $play//tei:titleStmt/tei:title
```

We use the `let` statement to set a variable `$plays` equal to the collection of all 42 plays. We then use a `for` statement to iterate over the plays, doing something once for each play. Remember that the components of a FLWOR expression must occur in FLWOR (for, let, where, order by, return) order, except that you can have as many `for` and `let` statements as you want, and they can occur in any order with respect to one another. If there is a `where` statement, it must follow all of the `for` and `let` statements; if there is an `order by` statement, it must come next; and the `return` statement must come last. There must be at least one `for` or `let` statement and exactly one `return` statement; everything else is optional.

While we iterate over the plays with our `for` statement, we use an `order by` statement to sort them alphabetically, taking advantage of the fact that the first textual content of each play is the title, so alphabetizing by the entire text of the play is equivalent to alphabetizing by the title. Note that when we return the title of each play, the XPath in the `return` statement has to begin with the variable `$play`, so that on each of the 42 iterations we'll be getting the title of the play we're looking at at that moment.

The texts

Obdurodon contains the text of forty-two Shakespearean plays. In the TEI markup for these plays, speeches are `<sp>` elements and the speakers are `<speaker>` elements that are their first children. For example:

```
<sp who="Roderigo">
    <speaker>Roderigo</speaker>
    <l xml:id="sha-oth101040F" n="40">I would not follow him then.</l>
</sp>
```

There are 966 distinct speaker names (values of the `<speaker>` element) in the entire corpus, some of which show up in more than one play. Three of the names show up in more than ten plays: there is a character called “Messenger” in 22 plays, one called “All” in 20 plays, and one called “Servant” in 18. (These aren’t the same messenger or all or servant, of course!) Here are the details:

- Messenger appears in 22 plays: Antony and Cleopatra; Coriolanus; Cymbeline; Hamlet, Prince of Denmark; Julius Caesar; King Lear; Macbeth; Much Ado About Nothing; Othello, the Moor of Venice; Pericles, Prince of Tyre; The First Part of King Henry the Fourth; The First Part of King Henry the Sixth; The Life and Death of King John; The Life of King Henry the Eighth; The Merchant of Venice; The Second Part of King Henry the Fourth; The Second Part of King Henry the Sixth; The Taming of the Shrew; The Third Part of King Henry the Sixth; The Tragedy of King Richard the Third; Timon of Athens; Titus Andronicus
- All appears in 20 plays: A Midsummer Night’s Dream; All’s Well That Ends Well; Antony and Cleopatra; As You Like It; Coriolanus; Cymbeline; Hamlet, Prince of Denmark; Julius Caesar; Macbeth; Pericles, Prince of Tyre; The First Part of King Henry the Sixth; The Life of King Henry the Eighth; The Life of King Henry the Fifth; The Merry Wives of Windsor; The Second Part of King Henry the Sixth; The Taming of the Shrew; The Third Part of King Henry the Sixth; The Tragedy of King Richard the Second; Timon of Athens; Titus Andronicus
- Servant appears in 18 plays: All’s Well That Ends Well; Hamlet, Prince of Denmark; Julius Caesar; Macbeth; Measure for Measure; Pericles, Prince of Tyre; Romeo and Juliet; The Comedy of Errors; The First Part of King Henry the Fourth; The First Part of King Henry the Sixth; The Life of King Henry the Eighth; The Merry Wives of Windsor; The Second Part of King Henry the Sixth; The Taming of the Shrew; The Tragedy of King Richard the Second; The Winter’s Tale; Timon of Athens; Twelfth Night or What You Will

Assignment

Your assignment is to write XQuery that will query the collection of plays, find the three `<speaker>` element values that occur in more than ten plays, and return a list of those elements along with the names of the plays in which they occur. That is, you should write XQuery that generates the basic information in the preceding list. Once your XQuery is working, copy it into a plain-text document (not a Word document), save it, and upload it to CourseWeb. You can create a plain-text (that is, “.txt”) document in <oXygen/> by creating a new document of type “Text”. We don’t need to see the output of your XQuery; all we need is the XQuery code itself.

Our list includes some enhancements that you can consider optional for this assignment, but if you get the basic solution, we hope you’ll try them:

- We’ve added some plain text (“ appears in ”, “ plays: ”) and punctuation (the list of play titles is preceded by a colon and the individual titles are separated by semicolons). Whether you use this specific formatting is optional, but you have to use some sort of formatting to produce a readable list. An alternative might be to output HTML nested lists, where the outer list gives the character names, with the frequency values after the names in parentheses, and the sublists under each name contain the titles of the relevant plays. The formatting is up to you, as long as it's clear, legible, and sensible.

- We've sorted the results in descending order of frequency, so that the character name that appears 22 times is first, then the one that appears 20 times, then the one that appears 18 times.
- We've alphabetized the titles of the play. We had to do that explicitly in our XQuery; by default they come out in no particular order (well, there is a particular order, but it's the order in which they were uploaded into eXist, so it isn't meaningful to humans).
- The XML uses straight apostrophes in the titles, but we've replaced them with curly (typographic) ones.

Our solution used several `let` statements to configure "convenience" variables. We used `distinct-values()` and a `for` statement to find and iterate over all of the distinct `<speaker>` names in the collection. We used the `count()` function and a `where` statement to find all of the names of speakers that occur in more than ten plays. Finally, we used both `concat()` and `string-join()` in our `return` statement to glue the pieces together. For the optional parts, we used an `order by` statement to sort the results in descending order of frequency, and to alphabetize the play titles we used an embedded FLWOR statement. That means that we set a variable equal to the return of a FLWOR statement, which is a powerful way of using FLWOR not just for the main program flow, but also to control subcomponents. We used the XPath `translate()` function to fix the apostrophe.

This is not a difficult problem intellectually: you can paraphrase it easily as "find all of the speaker names that show up in more than ten plays." It is, however, a complicated pattern to formalize, and because the XQuery version has to be explicit, it is longer than the prose formulation. We would suggest breaking the problem down into subcomponents like

- How would you find all of the plays in the corpus?
- How would you find the title of a play, or of all of the plays in the corpus?
- How would you find all all of the speakers (that is, all of the <speaker> elements) in a play or in the corpus?
- How would you find all of the *distinct* speakers in a play or in the corpus?
- How would you find all the plays in the corpus in which a speaker with a certain name appears, and then count them?
- How would you find all the plays in the corpus in which a speaker with a certain name appears, and then list their titles?
- If you can count the number of plays in which each distinct speaker name occurs in the corpus, how can you use that information to sort your results in descending order by frequency?
- If you can find the titles of the plays in which a particular speaker name appears, how can you sort them alphabetically?

Some of the preceding questions are easier than others, but getting all the way to a solution means getting them all correct and stitching them together effectively, which is why we recommend developing your XQuery step by step and testing it after every step, so that when it breaks, you'll know which statement caused the problem. If you can't get all the way to a solution, that's okay, but you can't just give up; what you need to do instead is submit your solutions to as many of the pieces as you can (both the ones we list above and any others that you recognize as important). While this particular assignment is a bit (okay, more than a bit) artificial, the techniques required are very common in Real Life: find information across documents and create a report based on an aggregated result.

Watch out for namespaces!

The plays are all in the TEI namespace. This means that you'll want to declare a namespace prefix and map it to the TEI namespace, as you did in the last assignment. What tripped up several people in the last assignment, though, is that *all* element names in your XPath expressions require the namespace prefix. For example, if you've set the value of a variable `$plays` to the 42 play documents in the corpus and if you've also set the prefix `tei:` to point to the TEI namespace, you can get the titles of all of the plays with the XPath expression `$plays//tei:titleStmt/tei:title`. If you forget the prefix on either `<titleStmt>` or `<title>`, though, you won't get any results because you'll be looking for one or both of those elements in no namespace, and every element in this corpus is in the TEI namespace.