newtFire {dhlds}
Maintained by: Elisa E. Beshero-Bondar (ebb8 at pitt.edu) Last modified: Monday, 19-Feb-2018 17:57:25 EST. Powered by firebellies.

XQuery Exercise 2

Our second XQuery homework exercise works with our Digital Mitford project files, coded in TEI. Here's how to locate this collection in our eXist at the Pittsburgh Supercomputing Center: `collection('/db/mitford').`

IMPORTANT: You'll need to **declare the TEI namespace** at the top of any XQuery you run on these TEI files or you won't see any output. (Declare the namespace in the line just under `xquery version "3.1";`). Here's the line you need for that:

declare default element namespace "http://www.tei-c.org/ns/1.0";

Write XQuery expressions for each of the following tasks. Paste these into a text file or a document, and where we ask you to do so, record your return or output. Save your file using our standard homework filenaming conventions (as in `besheroXQuery2.txt`), and upload your assignment to Courseweb.

1. First, let's do some exploring. This is a large collection of files, so we might want to look at a complete list of their file names and get a count of the number of files. For this, we use the XPath function `base-uri()`. Write a statement in XQuery that returns the base-uri() (or filename) for each file in the Mitford collection. How many files are there? (Record the number.) Wrap the expression in a `count()` function so you return the number, and record the expression you used.
2. Starting from the `collection()`, write a basic XQuery expression to show you the coding of the files, using `/*`, so that you can see how to locate the `<title>` element inside the `<teiHeader>` and `<titleStmt>`. Copy this into your text file recording this homework exercise.
3. Begin working on FLWOR expressions. First, write a very simple FLWOR statement to define variables that will return the following:
    - the whole collection
    - the particular texts in the collection, starting from the `<body>` element in our TEI files. (You'll need this later.)
    - the main title of the files (as described in #2: up in the `<teiHeader>`, inside the `<titleStmt>`).
    - Write a return statement to return the text ONLY of the main titles of the files in this collection. Refer to what you learned in XQuery Exercise 1 about the differences between text() vs. string(). Which one of these should you use here and why? (Copy your FLWOR into your text file for this homework exercise, together with your explanation.)
    - How many titles did you return? (Record the number.) Note: You should actually return one extra title compared to the number of file names you returned, which surprised us as we worked with the collection, until we realized that one file here actually has *two* `<title>` elements inside its `<titleStmt>`.
    - **Bonus:** In Real Life we would write some more XQuery to figure out what is going on when we receive a puzzling result like this. To find out which file is the culprit, with two titles instead of one, we wrote a little more XQuery code, using a "for loop" and another variable to return the file in question. See if you can write the code to locate that file for a bonus on this assignment. Record your XQuery and give the two titles of the file.

4. Build on your FLWOR expression. We are looking for some very important personal contacts of Mitford whose names turn up in the `//body` of **more than 15 files** in the Digital Mitford collection. Note:
    - We don't want results from the TEI Header because that would include the current Mitford editors, so we want to define our variable with an XPath that drills into the `body` element of each file.
    - Each person is coded with a distinct identifier held in an @ref attribute: `<persName ref="#id">`, which helps us to keep track of people when they are referred to by different names in the archive.
    - Define a variable to collect a list of the distinct values of this `@ref` attribute. **Note:** To get output in XQuery when you collect a list of attribute values, you need to return the `string()` value of the attribute. (Check your results here. We returned 711 distinct values of the `@ref` attribute on the `<persName>` element across the collection.)
    - What we need to do next is something like building an index in a book. We have isolated the 711 distinct values of all the identifiers for people across the collection. We now need a way to check and see if a file contains a `<persName @ref>` that matches each single entry in that list of distinct values, since those entries are now just a simple *dereferenced* list, and pulled out of their XPath context. A good example of what we are doing is the index of a printed book: You go to the back of a book and look up a word or phrase that is only listed once, and find out all the different places in the body of the book that mention it, so you can flip back and find what you need. What we are doing here is similar: We want to define a new variable in XQuery to look up for us *which files are holding each entry* on our list of distinct values. We'll need to test each entry in our list of distinct values (using a "**for loop**") and *map it back into the XML tree* to locate the files holding it. You will need a predicate expression that uses a comparison operator (and we used a General Comparison operator, the equal sign = to filter *which files contain `<persName>` elements with `@ref` attributes* ***equal to*** *the current distinct value in the "for loop"*. Here is a review of comparison operators, which you used in the XPath assignments.
    - Now, use the `where` statement in the FLWOR to filter your results so that you return *only the distinct `@ref` attributes that are seen in more than 15 files*. **Note:** This make take about 15 seconds to run, so do not be alarmed if eXist seems to pause a little while. Our collection of files is pretty large and you are testing a list of 711 distinct values in your "for loop"! When the dust settles, you should return a list of 10 name ids, a "top 10" list of popular names in the Digital Mitford collection.
    - We'd like to return the output without the hashtag (#) in front, and we want to output the results in alphabetical order by the `@ref` (without hashtag). To eliminate the hashtag, we recommend either the `tokenize()` function we used in the previous assignment, or the `translate()` function (read about these in Michael Kay or look it up in The XPath Functions We Use the Most.
    - Reading our Explain XQuery Guide to look up the details, write the FLOWR statement to `order by` the distinct @ref that is most frequently referenced.
    - Add the appropriate word to the "order" statement to generate these results in reverse order. (Refer to our guide linked here.)

5. Finally, we will build an HTML file around your XQuery results using curly braces `{ }` where necessary. Consult our **XQuery Intro Guide on building HTML with Curly Braces.** We need to make some changes to our file, though, because we are working with two different namespaces now, HTML and TEI. Alter the top lines of the XQuery so they look like this:

```
            xquery version "3.1";
declare default element namespace "http://www.w3.org/1999/xhtml";
declare namespace tei="http://www.tei-c.org/ns/1.0";
```

   And begin building your HTML around your FLWOR so the basic outermost structure of the file is set:

```
   xquery version "3.1";
declare default element namespace "http://www.w3.org/1999/xhtml";
declare namespace tei="http://www.tei-c.org/ns/1.0";
  <html>
<head><title>Top Ten Most Referenced People in the Digital Mitford Project</title></head>
<body>


{
. . . FLWOR HERE. . .
[use the tei:prefix on any TEI element names here]
Return $something

}
</body>
</html>
```

6. Build an HTML table in the HTML body part of the file to contain table rows and two columns of cells (two cells side by side in each column).
    - The first cell will contain each of the top ten the translated and sorted `@ref` value (without hashtag) that you retrieved in number 4.
    - In the other cell, return a string-joined list of the `base-uri()` or filenames of each file that contains the match, separated by commas. You might just want to trim down that `base-uri()` so you only return the filename at the end (like we did in XQuery Exercise 1. Here is our HTML table output to view in a web browser. (View Page Source to look under the hood at the HTML code.)
7. **Bonus**: Instead of using `string-join()` to list out the multiple filenames in your second table cell, can you work out how to output that list of names in its own table, nested inside that second table cell? Inside the table cell, you will need to nest a new FLWOR statement inside a new set of curly braces, in which you'll make another `for` statement. Return your output in table rows with a single column of cells.

Copy your HTML and FLWOR constructions into your document to upload to Courseweb.