




[newtFire {dhlds}](#).

Maintained by: Elisa E. Beshero-Bondar (ebb8 at pitt.edu)  Last modified: Sunday, 22-Oct-2017 21:10:17 EDT. [Powered by firebellies](#).

---

## XSLT Exercise 2

### Overview of the Assignment

The Digital Mitford Site Index stores lists of names and information on people, places, organizations, and texts, among other kinds of named entities referenced in files throughout the Digital Mitford project. For this assignment, we will work with a slightly modified version of the Digital Mitford Site Index, which you should download [from here](#) and open in `oXygen`. Our goal is to create a structured outline in HTML of all the information about organizations in the site index. We want to output that in HTML in the form of a list with nested lists inside, representing an outline of first the *categories* of organization, and then inside each category, a new list of the organization names. This is something we actually need to do in the Mitford project: to process portions of the Site Index file to make it readable on the web as a list. One possible use of a webpage like this is as a list of links, so that each organization name might link to a page of information on each organization. We don't have to generate those links now. For this assignment, we just want to learn how to transform XSLT to HTML and to generate the lists themselves by pulling the right content out of our XML.

If you're feeling adventurous, once you obtain the output we're seeking, you may go on to build other HTML lists, working with other portions of the XML document, such as the `<listBibl>` or `<listPerson>` sections, which are formatted a little differently. The only required content of your homework, though, is the HTML outline of **organization types** and **organization names**. For the organization types or categories, we need to pull from the `<head>` element sitting inside at the top of each `<listOrg>` elements in our TEI file. For the organization names, we reach in to find the individual entries for `<org>` and their child `<orgName>` elements inside each `<listOrg>` element. Each `<org>` element contains one `<orgName>` inside that holds the best-known name of a particular organization. You may first want to experiment with XPath on the Site Index file to locate the `<listOrg>` elements and study the XML hierarchy of the lists. Let's make the outer list be **ordered** (or numbered) list in HTML, using the HTML `<ol>` element, and then make the inner list be an **unordered** (bulleted) list, using the HTML `<ul>` element.

Your lists in HTML should come out looking something like this, only yours will have a few more entries in each category, because your XML document contains some new material.

## 1. Archives Holding Mitford's Papers

- Baylor University, Armstrong Browning Library
- Berkshire Record Office
- British Library
- Boston Public Library
- Cambridge University: Fitzwilliam Museum
- Duke University Rubenstein Library
- Eton College
- Florida State University Special Collections
- The Women's Library, Glasgow
- Houghton Library, Harvard
- Huntington Library
- University of Iowa Special Collections
- Massachusetts Historical Society
- New York Public Library
- Oxford University, Balliol College Archives
- Oxford University, Bodleian Library
- Reading Central Library The principal archive of Mary Russell Mitford's personal papers and related documents, holding approximately 1,000 manuscripts and a nearly comprehensive collection of her publications.
- John Ruskin Library, Lancaster
- The John Rylands Library
- National Library of Scotland, Manuscript Collections
- University of Texas, Ransom Center
- University of Reading Special Collections
- University of Virginia Special Collections
- Wellesley College, Margaret Clapp Library, Special Collections
- Wordsworth Trust
- Yale University, Beineke Library

## 2. Organizations Relevant to Mitford's World

- Billiard Club
- House of Bourbon
- Cavaliers
- Court of Chancery
- Church of England
- the Cockney School
- Drover
- Eton College
- High Court of Justice
- House of Commons
- the Kembles
- House of Medici
- Mitford
- Mr. and Mrs. Mitford
- the Moncks, family of John Berkeley Monck
- New Model Army
- Palmerite
- Parliament
- Court of Pope Pius VII
- Prelacy
- the Presbyterian faction
- Privy Council
- Richmond Coach or Stage
- Scriblerus Club
- Slade family
- Taylor and Hessey (publishers)
- Tory Party
- Twickenham Coach or Stage
- Valpy family
- Webb family
- Weylandite

### 3. Fictional Organizations Referenced by Mitford

- Attendants &c.
- Citizens
- Guards
- Guards
- Ladies
- Nobles (in Julian)
- Nobles (in Rienzi)
- officers in Charles I
- Prelates

The underlying HTML, which we generated by running XSLT, should look like this:

```
<ol>
  <li>Archives Holding Mitford's Papers<ul>
    <li>Baylor University, Armstrong Browning Library</li>
    <li>Berkshire Record Office</li>
    <li>British Library</li>
    <li>Boston Public Library</li>
    <li>Cambridge University: Fitzwilliam Museum</li>
    <li>Duke University Rubenstein Library</li>
    <li>Eton College</li>
    <li>Florida State University Special Collections</li>
    <li>The Women's Library, Glasgow</li>
    <li>Houghton Library, Harvard</li>
    <li>Huntington Library</li>
    <li>University of Iowa Special Collections</li>
    <li>Massachusetts Historical Society</li>
    <li>New York Public Library</li>
    <li>Oxford University, Balliol College Archives</li>
    <li>Oxford University, Bodleian Library</li>
    <li>Reading Central Library The principal archive of Mary
      Russell Mitford's personal papers and related documents, holding
      approximately 1,000 manuscripts and a nearly comprehensive collection of her
      publications.
    </li>
    <li>John Ruskin Library, Lancaster</li>
    <li>The John Rylands Library</li>
    <li>National Library of Scotland, Manuscript Collections</li>
    <li>University of Texas, Ransom Center</li>
    <li>University of Reading Special Collections</li>
    <li>University of Virginia Special Collections</li>
    <li>Wellesley College, Margaret Clapp Library, Special Collections</li>
    <li>Wordsworth Trust</li>
    <li>Yale University, Beineke Library</li>
  </ul>
</li>
  <li>Organizations Relevant to Mitford's World<ul>
    <li>Billiard Club</li>
    <li>House of Bourbon</li>
    <li>Cavaliers</li>
    <li>Court of Chancery</li>
    <li>Church of England</li>
    <li>the Cockney School</li>
    <li>
      Drover
    </li>
    <li>Eton College</li>
    <li>High Court of Justice</li>
    <li>House of Commons</li>
    <li>the Kembles</li>
    <li>House of Medici</li>
    <li>
      Mitford
    </li>
    <li>Mr. and Mrs. Mitford</li>
    <li>the Moncks, family of John Berkeley
      Monck
    </li>
    <li>New Model Army</li>
    <li>Palmerite</li>
    <li>Parliament</li>
    <li>Court of Pope Pius VII</li>
  </ul>
</li>
</ol>
```

```

        <li>Prelacy</li>
        <li>the Presybterian faction</li>
        <li>Privy Council</li>
        <li>Richmond Coach or Stage</li>
        <li>Scriblerus Club</li>
        <li>
            Slade family
        </li>
        <li>Taylor and Hessey (publishers)</li>
        <li>Tory Party</li>
        <li>Twickenham Coach or Stage</li>
        <li>
            Valpy family
        </li>
        <li>
            Webb family
        </li>
        <li>Weylandite</li>
    </ul>
</li>
<li>Fictional Organizations Referenced by Mitford<ul>
    <li>Attendants & c.</li>
    <li>Citizens</li>
    <li>Guards</li>
    <li>Guards</li>
    <li>Ladies</li>
    <li>Nobles (in Julian)</li>
    <li>Nobles (in Rienzi)</li>
    <li>officers in Charles I
    </li>
    <li>Prelates</li>
</ul>
</li>
</ol>

```

In HTML ordered and unordered lists, the only elements permitted inside are list items or `<li>` elements. We've nested them so that each list item in the outside numbered list contains a category type (designating what kind of organization), followed by an embedded `<ul>` that contains, in turn, a separated bulleted list series, listing the name of each organization in the list.

## Before You Begin: Set up the XSLT Stylesheet to Read TEI

The Digital Mitford's Site Index file is coded in the TEI namespace, which means that your XSLT stylesheet (much as in the last assignment) requires an instruction at the top to specify that when it tries to match elements, it needs to match them in the TEI namespace. (When you create a new XSLT document in `<oxyen>` it won't contain that instruction by default, so whenever you are working with TEI you need to add it (See the text in [blue](#) below). To ensure that the output would be in the XHTML namespace, we added a default namespace declaration (in [purple](#) below). To output the required DOCTYPE declaration, we also created `<xsl:output>` element as the first child of our root `<xsl:stylesheet>` element (in [green](#) below), and we needed to include an attribute there to omit the default XML declaration because if we output it that XML line in our XHTML output, it will not produce valid HTML with the w3C and might produce quirky problems with rendering in various web browsers. So, our modified stylesheet template and `xsl:output` line is this, and you should copy this into your stylesheet:

```

<?xml version="1.0" encoding="UTF-8"?>
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
    xmlns="http://www.w3.org/1999/xhtml"
    xpath-default-namespace="http://www.tei-c.org/ns/1.0">

    <xsl:output method="xhtml" encoding="utf-8" doctype-system="about:legacy-compact"
      omit-xml-declaration="yes"/>

  </xsl:stylesheet>

```

## Guide to Approaching the Problem

Our XSLT transformation (after all this housekeeping) has three template rules:

1. We have a template rule for the **document node** (`<xsl:template match="/">`), in which we create the basic HTML file structure: the `<html>` element, `<head>` and its contents, and `<body>`—anything that appears just once in the HTML document (one to one relationship with the root node). Inside the `<body>` element that we're creating, we use `<xsl:apply-templates>` and select the `<listOrg>` elements (using an XPath expression as the value of the `@select` attribute). And we create our wrapper `<ol>` tags to set up the ordered list of organization types.
2. We have a separate template rule that matches the `<listOrg>` elements (holding the lists of organizations), so it will be invoked as a result of the preceding `<xsl:apply-templates>` instruction, and will fire once for each `<listOrg>` element in our Site Index. Inside that template rule we create a new list item (`<li>`) for the particular `<listOrg>` being processed and inside the tags for that new list item we do two things. First, we apply templates to the `<head>` for the `<listOrg>`, which will cause its category description to be output when we run the transformation. Second, we create wrapper `<ul>` tags for the nested list that will contain the names of the organizations within that category. Inside that new `<ul>` element, we use an `<xsl:apply-templates>` rule to apply templates to (that is, to process) the `<org>` elements of that `<listOrg>`.
3. We have a separate template rule that matches the `<org>` elements, which make up the items in the list of organizations, and that just applies templates to the `<orgName>` element within each `<org>`. This rule will fire once for each `<org>` element inside the `<listOrg>`, and it will be called separately for the `<org>` elements within each `<listOrg>`, so that the orgs will be rendered properly in their respective lists.

We don't need a template rule for the `<head>` elements themselves because the built-in (default) template rule in XSLT for an *element* that doesn't have an explicit, specified rule is just to apply templates to its children. The only child of the `<head>` elements is a text node, and the built-in rule for *text nodes* is to output them literally. In other words, if you apply templates to `<head>` and you don't have a template rule that matches that element, ultimately the transformation will just output the textual content of the head, that is, the title that you want.

## Important

- Those who like to read ahead or already have some programming experience with other languages may have noticed that XSLT includes an `<xsl:for-each>` instruction that *could* be used to solve this problem. *We are prohibiting its use for now*; your solution must use `<xsl:template>` and `<xsl:apply-templates>` rules instead. There's a Good Reason for this, which we'll explain later, when we talk about situations where you *should* use `<xsl:for-each>`.
- You may notice that two or three of your output bulleted list items show multiple related organization names squished together. This is because our editors occasionally provided more than one name used for an organization. Our standing rule is that the most definitive `orgName` be listed *first* in the list of names, so we recommend that you tidy up your list by selecting just the very first available `orgName`, that is, the first element child named `orgName` of `org` elements you are processing. Alternatively, you may try applying an XPath `string-join()` function to output the entries, but you will need to use `xsl:value-of` instead of `xsl:apply-templates` because we need to use `xsl:value-of` to calculate the results of functions (which removes us from the XML tree). Either approach is fine with us, and you would use the same `@select` attribute to indicate what you would like to output.
- *Before submitting your homework, you must run the transformation at home* to make sure the results are what you expect them to be. Remember, there's a guide to running XSLT transformations inside `<oXygen/>` in our [Intro to XSLT tutorial](#). If you don't get the results you expect and can't figure out what you're doing wrong, remember that you can post a query to our [DHClass-Hub Issues board](#). You can't just ask for the answer, though; you need to describe what you tried, what you expected, what you got, and what you think the problem is. We often find, just as we're preparing to post our own queries to coding discussion boards, that having to write up a description of the problem helps us think it through and solve it ourselves. We're also encouraging you to discuss the homework on the discussion boards because that's also helpful for the person who responds. Answering someone else's inquiry and troubleshooting someone else's problem often helps us clarify matters for ourselves!
- When you complete this assignment, submit your XSLT file and your output HTML file to Courseweb, following our usual homework file-naming conventions.