Ryan Ballenger
Final Project Report

Note: The application is titled assignment#2 in exist-db because that framework was used to start the final project. The homepage, which lists the departments, can be accessed with the following URL: http://localhost:8080/exist/apps/cscie18-assignment2/

Search

Search enables the user to find precisely what courses they need. First, the data in the search form is generated. All the distinct department names are found with xquery in the process.xq file and stored in <dname> elements. Similarly all the unique course group, terms, course start times, days, and professors are found and stored. The form data is then displayed via the XSL file project.xsl in select elements in a search form. The user can create a search by selecting one of the available, distinct values that was found. For example, the user can search for courses in the General Education department, in the Cultural Belief course group, during the fall term, at a specific time of day and with a specific professor. The user can also not specify one of the fields and leave the select input as 'All' which is a term that the Xquery file process.xq matches to all the courses. The user is also given an option to enter a phrase which searches the course title and course description for matches. If search criteria is entered for any field, the search is exclusive meaning that all matching courses will include all the search criteria that is specified.

The advantage to this search design is that the user is given access to all the available values for the search fields. Through an extensive select input that contains all the departments or course groups or other fields, which are generated ahead of time, the user knows the options for the search fields.

The processing of the search occurs in the process.xq file as well. The search terms are passed as parameters that are stored as variables in the Xquery file. All the courses are then found that match the search criteria. This is done in the "where" part of a query at the bottom of the process.xq file. If search criteria is available for a field, it is applied and otherwise, true() is applied to include all the courses. Data on all the matching courses is then sent to the project.xsl file to be displayed in a table.

Sort

The course matches are sorted by default by the course department, course group, and short title. The user is also given the option, after a search, to sort by the term, the course type, the start time, and the meeting days. This is done by checking the corresponding boxes and clicking submit.

The precedence of the ordering is the term first because if a user wants to sort by the term, they are probably only interested in classes during a single term (e.g. the spring). Next, sorting is done by course type, if this sort attribute is requested, since the user probably only wants a certain course format. Lastly, sorting by day and by time is applied, if requested.

Sort works by storing all the current search's data in a form in hidden fields. When the user clicks sort, the same search is performed, by passing the search criteria in as parameters again, and the matches are sorted accordingly. To confirm the same search was performed and only sorted differently, the search criteria is displayed which is also useful to remind the user of their search criteria.

More specifically, the sorting occurs in process.xq near the bottom of the file. If the user checked a sort criteria, the corresponding parameter will exist, namely $sterm, $stype, $stime, etc. If the parameters exist, ordering by that parameter is applied by the precedence described earlier to all the courses that match the search.

Navigation

The user is always given the same heading with links to departments, course groups, and search. This enables him or her to be able to begin a new search quickly or go back to the home screen.

After a search is performed, the search criteria is listed for the user to know what results they are given. This is a navigation feature because it tells the user what page they are on, which is the page generated by the search they have entered. Since there are many search fields, it would likely be helpful for the user to see what

search they specified.

Another navigation feature is the page number listed by the next and previous page links. Since only 20 courses are displayed per page, this serves to tell the user where they are in the results. The page number is passed as a parameter, used to determine which part of the subsequence to display, and displayed at bottom of the page to tell the user where they are in the results.

PDF's

Every page on the site has a link to create a pdf version. For course searches, the search and sort criteria is passed to the Xquery file courselistpdf.xq, where a pdf is created for all the matching course data. Besides the course search page, the other pages are the departments and course group listings, which pass their data to the Xquery file other-listing-fo.xq to make a pdf of those lists. For all the course listings pdfs, a table of contents is made to show the included courses and link to the more detailed information about each course.

Mobile

A mobile version of the site is generated after the screen width is less than 850px.

Less data: Less data in the mobile version enables it to load quicker, in a mobile computer that has a lesser internet connection. In the mobile version, the Harvard emblem and name are no longer displayed. The mobile site begins with the navigation menu including departments, course groups, and search. The footer is also no longer displayed, since it is informative but not necessary. The data about each course is also reduced to the necessary values including the short title, the title, and the professor in the results page. Similarly in the single course view, 5 of the 7 course fields are now shown to give the user the relevant course information without overwhelming a mobile device.

Reformatting: In addition to less data, the pages are reformatted in the mobile version for better access. The navigation menu is vertical which is more convenient for scrolling and clicking on a mobile device. In the single course view, the description is moved below the other fields in the mobile version. Moving the description also makes for a more vertically-aligned mobile version. The length of the page is not changed because 20 course results is relatively small and not overwhelming for a mobile device.

As discussed by Google's "What makes a good mobile site?", a link to search is always available to the user. It is located in the navigation menu as the third item in every page. Since mobile pages can be more difficult to navigate, this enables the user to quickly perform a new search if they haven't found what they need. Overall, the navigation in the mobile version is simple and easily accessible with big buttons at the top of the page, which is convenient for a mobile user who can see less of a page at once.

Validation

The pages are all valid in HTML5 and validated by validator.w3.org.

Reflection

What worked well

The search format is well done. I considered using manual entry for search fields and also for JavaScript that would give the user suggestions as they typed. The guided search process of giving the user the available terms for each field in select inputs seems to give the user relevant information as they create a search.

The forms with hidden data to make the links, including the next and previous page links, the sort link, and the pdf view link, also worked well. The data for the current search needs to be passed into these links to be processed. For example, the sort link needs access to what search has happened to sort those results. By storing the current search and sometimes sort data in hidden fields in these forms, each link is able to effectively work with the current state of the program. I had considered concatenating a URL with the parameters needed or processing the data ahead of time, which would not have worked as well.

I also utilized the search page that I built when doing the department and course group pages. When the user clicks on a department, it links to a search for that department. I could have created separate pages to generate data for the courses in the departments or course groups, but I prevented repeating myself by funneling the processing to the search page.

Lastly, I think the mobile version of my site turned out well. It gives the same search tools to the user but in a simpler, mobile-friendly way. The vertical navigation menu and the reduced course data tell the user what he or she needs without giving them too much data. It also has the appearance of some of the popular mobile sites today from a user perspective. The functionality of the search is not reduced, allowing the user to search as precisely as on a desktop, which is a good component to preserve in the mobile version.

Less well done and lessons learned

A few instances of data processing could have been improved. For example, I should have combined the days and times at the beginning instead of trying to handle both separately. In the single course view, I used this method and it worked well for transferring and displaying the days and times of class meetings. In the course search view however I spent too much time and code handling these values.

Also, if I did the project again, I would always use mixed XML data types from Xquery processing. It allows for precise data selection as in getting the @term_code for a course. This strategy is better and enables more control compared to using an outer element with many elements contained within.

A lesson learned is that error-checking should be done after each major step in the project. Finding an error at the end makes it difficult to determine where the processing error is occurring. With thorough testing after each improvement, an error or bug is more likely to be caught and fixed when it occurs. I was able to find and fix my errors, but in retrospect, I wish I would have performed more testing and caught them before they were difficult to find.

I also learned that displaying data is helpful. In the past, I have assumed that I know or can figure out the data that is being processed. However, bug checking will be required, and if mechanisms are built to display all the current search and sort data, it is easy to locate where the error is or confirm the program is working properly. For example, by displaying all the data that would be sent to the next and previous page links, I confirmed they were doing the right search and going to the correct page number after being clicked.

XML Technology Implementation

Data and presentational aspects are separated. They collaborate well and the data is formatted such that it can be easily displayed. The data processing happens in the Xquery files, is completed, and the XSL files simply make the web pages to show the data.

Structures are flexible and adapt to differently-formatted data. For example, some classes have more than one meeting day and time which the program is built to handle. I took all the possibilities into consideration and enabled my program to be able to adapt to a wide-range of situations.

I used xsl:import to apply common formatting to all the pages. Each page has the navigation menu, the title and picture, and the footer which are held in the common.xsl file. Also the first step in the apply temple matching occurs in common.xsl.

XQuery is structured for DRY. I was creating multiple search results pages including a version for when the search link is clicked and one for displaying a search. However, I found a way to combine these Xquery files into one file process.xq along with eliminating redundant steps. In one case, I was passing the same criteria through an Xquery element. I was able to create an Xquery function to produce the element values and prevent needing to do this step twice. The Xquery function supports the DRY strategy.

New departments would not require any code changes. Since the search field values are generated dynamically, it would show up as a search option in the department select input of all the departments to search. Similarly, new data fields for the course would require small adjustments. The only change would be to add it as a course attribute to be passed to the xsl file. Also, expanding this for a University catalogue would not be difficult because it does generate data dynamically. It is designed based on knowing the department, course group, and other data points about a course, which would apply to a University-wide course catalogue example. Searches can be very specific since 7 fields are available which enables a student to find what they need even when the catalogue is expanded. Overall the site is designed for general use and is not customized or built for this particular FAS group of courses.

Extraordinary Distinction

Some of my project features, designs, and implementations deserve extraordinary distinction.

When creating pdf's of sorted course results, I maintained the ordering of the current sort in the pdf. The first version of the site made a pdf of the unsorted list of courses meaning it contained all the courses for the current search but did not have sorted ordering (e.g. by term and by type). The final version of the site passes data on what sorts of taking place and implements that sort into the pdf version. For every possible search and sort, there is a pdf equivalent that exactly matches the results.

As mentioned in earlier sections, I believe that generating all available search options during the page load and giving this data to the user in select-style search inputs deserves extraordinary distinction. Typically searches are based on user-generated input but can difficult to use because the user does not know what he or she wants. By quickly generating all the values of a search field (e.g. all the possible departments), the user is given a select input that communicates to them what departments they can search for. The possible values of the search field can be generated quickly with a distinct-values function in the Xquery file which means the initial search page can still load rapidly too.

As we learned in assignment #4, I used mixed XML content generated by the Xquery files as much as possible. At first, a few elements containing elements were used, but largely mixed content, as in elements containing attributes, are used to pass data to the XSL files. Mixed content uses less memory and can be processed more quickly. Mixed content is superior to only attributes or only elements which was highlighted in lecture. Specifically, the data passed to make PDF's, the course info data, the next and previous page data, and the professor data is stored using mixed content and well-styled XML.

Another good design choice was converting to mobile based on page width. The first versions of this site used a different URL for the mobile versions but this caused repeating code segments. It is more dynamic and prevents repeating code to use a CSS style sheet that applies once a certain page width is reached. From a developer's perspective, this enables testing the mobile site to be as easy as changing the browser width instead of working with multiple pages. Code updates do not need to be applied to a mobile and desktop XSL file and can be applied once to the dynamic XSL file that adjust to mobile. From the user's perspective, it is useful too to have the mobile version available on the desktop and not be required to switch URLs for the mobile version. If the user narrows a window, to fit multiple applications on their screen, the mobile version appears enabling them to conveniently navigate the site in a smaller area.

Lastly, a strength of my search process is disabling the data from the previous search, which is not necessary and was not implemented in earlier versions. As mentioned, it is useful to the user, when there are many search fields, to remind them about the specific search they ran. For the developer, it is productive to see that the search is being handled properly and the results apply to the designated search. This serves as a good source of navigation for the user too. Details about the search results site page are given to them directly to show them what the course results list means. If they were to minimize a page and return to it later, they would be able to see the search criteria that is currently being used. Since the site is designed to perform searches, it is very helpful to have this feedback loop and show the search criteria on the search results page.