CSCI E-18
Final Project
David Glauner

Sites and Content:

The site is laid out to facilitate easy searching and browsing for the user. There are two main channels to go down from the landing page. The user can locate courses by browsing by department and then course group. Or the user can search with a list of custom criteria of many options, including a keyword search. The site is XHTML 1.1 strict compliant.

To handle the mobile requirement, I used the responsive CSS framework Bootstrap. With some very basic setup, it made the site scale very well, up and down. I purposely gave the site a more simple layout so it would scale easily. Some grid columns get hidden on small devices, to make them fit the screen better. To help users on some of the longer pages, i.e. department browse and search results, pagination has been added so as to decrease the load time over a potentially slow connection.

PDF generation is accomplished on the department browse channel by either a comprehensive department listing of classes, or individual class PDF's. On the search channel, you can generate individual class PDF's from your search results.

XML Technology Implementation:

Source data is filtered by and comes from a BaseX database. I loaded the xml file into the database. There are 2 main xquery queries setup as resources in the sitemap. These two queries supply filtered data for all the views on the site. Filtering data requests are supplied by querystring, which then gets passed on through to the BaseX database, and parsed into the query. For example, when you view a page for a single course, the database is generating an XML file with only one course in it. To facilitate pagination on the department browse page, there is a departments query, which spits out a special XML file of unique departments, built off the course catalog. In theory then, the XSL transformation pages don't have to do too much heavy lifting, and just display what the database sends them.

The search page is very flexible all the drop-down lists (except days of the week) are populated off of the course catalog, so when the next years catalog comes out, the search page won't require any changes. New departments, different course meeting times, etc. would pose no problem for the site.

Since the data filtering (database) is separate from the XSL transformations, it should be relatively easy to add other output formats.

Common elements, templates, etc. are on a common XSL page that is imported into all the other pages. For the PDF output, there is also a common PDF page that the other PDF pages use.

Page output is generated on the fly with XSL files and Cocoon.

Extraordinary Distinction:

I am really pleased with the implementation of the queries on the BaseX database. And since the main query was so straight forward, I could include it as the data-source for every page. Also the second query/filter (departments) made paginating the department listing very easy.

Lesson's learned:

Time... I didn't have enough. There are more changes I would have loved to do for several of the pages, but I just ran out of time...

Running the application:

I implemented the BaseX database exactly as demonstrated in class and section. So if you copy the xq directory from the project folder to the BaseX/Webapp folder and load the courses xml as a database called “courses” you should be up and running. If you already have the courses database setup under a different name, then you just need to update that the resources in in my sitemap file to match your database name.