CSCU9T4 XML Assignment Report

# Schema Choices:

First of all, for the XSD, I decided instead of having a series of nested elements, to create multiple complexTypes. I created an element called questionList, which is a complexType and can contain multiple question elements. The question elements are as well complexTypes and have a sequence of nested elements; queries and answers. The query element is simply the content of each question, whereas the answer element, another complexType, can contain different answer choices. Also, a question can optionally have a comment.

The answer element can have a sequence of choices: Yes/No/Blank/Comment or Sit/Stand/Blank/Optional/Comment.

Finally, the comment element is a complexType itself, and contains an id attribute, a content and optionally another comment as well.

# CSV vs XML files:

There are numerous differences between an XML file and a CSV (comma separated values) file. First of all, XML is like HTML in that it is a universal language (although it differs from HTML in the fact that while HTML is a very well defined dialect, XML is actually a generic meta-language). A CSV file is a CSV file—meaningless except to the person using it.

A CSV is simply a flat table of data that doesn’t really support data hierarchies. An XML file, on the other hand, is intrinsically hierarchical—object-oriented, if you will, although you can still build relational information into it for specifying more complex relationships of data. Therefore, is very appropriate when receiving complex data as a response. It is also very human readable. Most browsers have built in XML readers that allow you to inspect XML files. Since XML was the first standard hierarchical data format, most APIs have built in functionality to automatically convert XML data streams into native data structures like objects.

Also attributes can be assigned to a given XML element beyond simply the data value of that element.

Furthermore, in order to transform data into some other form, a CSV file requires explicit programming. An XML file can use a special filter language called XSL (Extensible Stylesheet Language) to modify itself, and that filter can be swapped out for other filters with a single line of code.

Generally, though, CSV files are one third of the size of XML files, and in this way they help reduce bandwidth.

# Java implementation choices:

For the Java program, I used a DOM parser to create the XML file. What it does is, it creates an XML document as a tree structure, where every element of the XML is a node. Building a DOM representation, means that I could define the objects and properties of all XML elements and the methods to access them. As input I get the data from the surveydata.csv and the surveycomments.csv files and I make sure to do a validation to check if those files actually exist on the source folder. Then, I proceed to create scanners to go through every line of the csv files and save the data to appropriately named elements. To go through each line and separate the different types of data, I use a complex regular expression that splits each line wherever there’s a comma outside of double quotation. I decided to design such a regex, since the “,” delimiter isn’t effective, since there are commas within the content of data, apart from the standard comma separated values. Hence, I store each value into an array, and after I assign it accordingly, to the proper elements, that are created within the while loop for the scanner. I also make sure, give elements attributes, if any of the questions (or answers), contain comments. Moreover, I decided to create a static append() method that requires two elements as parameters and appends the second element parameter to the first parent element, after it checks that the child element isn’t empty in the first place. I also decided to create another static method that returns an array where the comments are stored, after reading the surveycomments.csv file. The parameters this method requires, is a choice integer (0 or 1) to either return an array containing the comment’s id or the comment’s content. I use the same regular expression I designed earlier, since there are other commas inside the comment’s content, apart from the standard comma separated values. Finally, I transform the XML file and automatically output it.

# XSLT Implementation choices:

For the XSLT files I decided to use the following XSLT Elements:

* XSL templates
* XSL Filters (for-each, value-of etc)
* XSL variables
* XSL sort
* XSL conditional ifs
* Methods such as position(), round()
* Groups <g>
* XML Graphics: SVG

As far as the fifth part of the assignment I decided to base mytransformation.xslt on the first XSLT and add more columns containing information about comments. Also I added a comment table where the comment ID and the comment content is being displayed. Lastly, I added a table containing other data that can be generated from the given XML file such as detailed total number of answers and the respective percentages.

# The code that was used:

I found that both the lectures and the practicals covered a big part of XML and its implementations, therefore nearly all the code that I used is based on that material. The only bits of code that I had to do an external research on, are methods such as round(), position(), how to use variables effectively (since they’re immutable) and the group <g> element. Also I had to use attributes such as order (for the sorting) and the transform attribute to manipulate text and rotate it, so that it’s outputted vertically.

As far as the Java code is concerned, again most of the code is based on lecture and practical material, apart from the use of regular expressions for the string delimiter, and the input file validation.

# Problems that were encountered:

Generally, there weren’t many difficulties I had to face for the assignment, given the fact that XML is a language that personally I haven’t had prior experience with. If I would have to state something that bothered me the most, that would be the difficulty of creating the bar chart, since making everything automated, required a rather thorough research and fully understand how many of the XSLT/XML element and methods work, since the use of variables is different, in comparison to languages like Java, where they’re mutable.

# Conclusion

To summarize everything, I found this particular assignment to be quite interesting and at the same time fun to complete. I got the chance to manipulate data and use structures that were previously unknown to me.