# *Web Programming V (420-C50-HR)*

# *Assignment 2 – XML Documents and Schemas*

Date assigned: Wednesday, October 4, 2017

Date due: **Wednesday, October 18, 2017**

**Learning Objectives**

Upon successful completion of this assignment, the student will be able to:

* Create XML Documents
* Work with Namespaces and XML Documents
* Define a schema for an XML Document
* Define a DTD for an XML Document

**Setup**

You can use any editor of choice, but you must tell me when you hand the assignment in what you used. I recommend XMLSpy

Create a new web application/website called *username*\_C50\_A01. Add the files in C50\_A01\_Files folder to your project.

To do:

**Part A – Creating XML Documents**

As may probably know, the Winter Olympics took place in Sochi, Russia early in 2014. We are going to look at some data from the participating countries in the last Olympics held in Vancouver, Canada in 2010.

1. Create a new XML file called countries.xml. Ensure that there is a prolog at the top of the document, indicating that this is an XML document using the UTF-8 encoding scheme and that it is a standalone document.
2. Copy the information from the countries.txt file to the countries.xml document.
3. Within the document’s prolog, insert a comment describing the purpose of the document. Include your name and the date in the comment text.
4. The countries.xml file should contain the following items:
   1. The root element of the document should be named countries. The countries element should contain multiple occurrences of a child element named country, each of which specifies details about a single participating country.
   2. Each country element should have child elements named name, athletes, medals and flagBearer.
   3. Each country element should also have a single **attribute** named “year” specifying the year the country first participated in the Olympic games.
   4. The name element stores the name of the country
   5. The athletes element stores the number of athletes attending.
   6. The medals element is required and has zero to three child elements. The child elements are gold, silver and bronze and contain the number of medals of that type that the country received. If the country did not receive any medals of that type, the element is omitted; however, the medals element is required and must always be there.
   7. The flagBearer element has two child elements. The lastName element contains the last name of the flag bearer and the firstName element contains the first name of the flag bearer.
   8. The sport attribute on the flag bearer element contains the name of the sport the flag bearer participates in.
5. Make sure to replace special characters like apostrophes (single quotes), accented characters and ampersands with the appropriate character reference or entity reference.
6. Create a processing instruction to attach the countries.css style sheet to this document.
7. Create a schema file for the XML file created. Call the schema file Olympics.xsd and set the target namespace to http://csdev.cegep-heritage.qc.ca/olympics.
   1. Make sure that the sport attribute comes from one of the 15 sports from those Olympics (see <https://en.wikipedia.org/wiki/2018_Winter_Olympics#Sports> for details). You may need to correct some errors.
   2. Link the schema file to the XML file and make sure that the XML file validates.

**Part B – XML Namespaces**

I have provided two stylesheets as a starting point for the Web page and the two XML vocabularies displayed in Figure 1.



Figure 1: Stars of the Month

1. Open the actor.css, movie.css, and stars.html and enter your name and the date in a comment section of each of these files.
2. Change the stars.html file into an XML document by adding an XML declaration to the top of the file and adding namespaces to the html element for the XHTML, actors, and movies namespaces.
   1. Set XHTML as the default namespace.
   2. Use a prefix of **act** for the actors’ namespace and a URI of http://csdev.cegep-heritage.qc.ca/actors.
   3. Use a prefix of **mov** and a URI of http://csdev.cegep-heritage.qc.ca/movies for the movies namespace.
   4. Within the head section of the HTML file, insert links to the actors.css and movies.css style sheets.
   5. Within the main div element, after the h1 and h3 heading elements, insert the elements from the actors.xml file. Place these elements in the actors’ namespace.
   6. Before each closing </Actor> tag, insert the movies associated with that actor from the movies.xml file. Place the movie elements in the movies namespace.
   7. Close the stars.html file, saving your changes.
3. Within the actors.css file and the movies.css file, place the style declarations in the actors’ namespace and the movies’ namespace respectively. Remember to use both formats. Close the files, saving your changes.
4. Open stars.html in your Web browser, verifying that the styles for the XHTML, actors, and movies vocabularies have been properly applied.

**Part C – XML Schemas**

1. Create a schema for the document client.xml document called client.xsd.
   1. Set the target, default and this namespace to http://csdev.cegep-heritage.qc.ca/client.
   2. Define a type for province, provinceType, which has as its values the two-letter abbreviations for each province (BC, AB, SK, MN, ON, QC, NB, NS, PE, NL).
   3. Create a postalCodeType that has the format X9X9X9.
   4. Define a percentType that has the range of values from 0 to less than or equal to 1.
   5. Define another complex type called clientType for the client element. Utilize the types you have already created (provinceType, postalCodeType, and percentType) for the sub-elements. percentType is used for discount. clientType has an attribute, clientId which is a positive integer.
   6. Define an element, **clients**, which consists of one or more client elements of type clientType.
   7. Use the schema to validate the clientSchema.xml document. You will have a couple of errors to correct before the file will validate. Comment out the DTD tag.
2. Create a schema for the document equipment.xml called equip.xsd.
   1. Set the target, default and this namespace to http://csdev.cegep-heritage.qc.ca/equipment.
   2. Create a type for the equipmentType called equipmentTypeValue, which should have only the values “PC”, “PR” or “SC”.
   3. Declare a named model group, equipmentGroup with elements equipmentName of type string, inserviceDate of type date, clientId of type integer and equipmentType of type equipmentTypeValue.
   4. Create a complex type, equipmentType, with the named model group, equipmentGroup and the attribute equipmentNumber of type positive integer.
   5. Define an element equipmentList which consists of one or more equipment elements of type equipmentType.
   6. Use the schema to validate equipSchema.xml. You will have a couple of errors to correct before the file will validate. Comment out the DTD tag.

**Part D – Olympic Schemas**

You have been assigned the task of creating a list of transaction for an invoice for your client. Look at the files invoice.xml and invoice2.xml. You may also want to look at the schema definition in invoice.xsd.

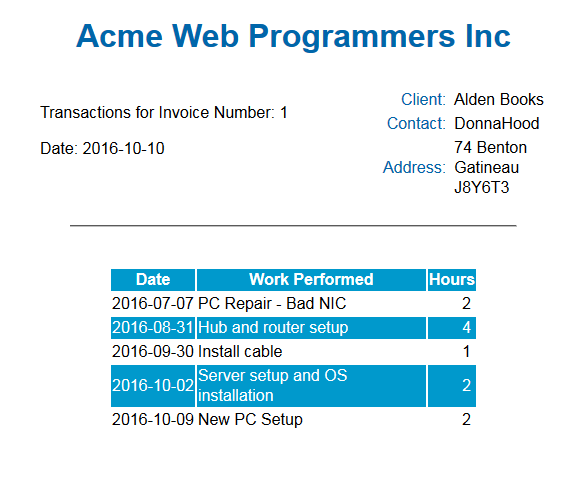
The invoice.xml file has the root element “invoice” and three separate complex elements.

* 1. <info> which contains information about the invoice including the date and the bill rate in dollars.
  2. <client> which contains the client information including their discount rate
  3. <work\_record> which can occur many times and contains a separate expense for each work item.

Create an XSL file which will transform the XML file to an HTML file which looks like an invoice that could be sent to a client based on the following requirements.

* 1. The top section of the page must contain the invoice number and date as well as the company name (Acme Web Programmers Inc).
  2. The next section must contain the Client information nicely formatted including all fields of the client element except the client\_id attribute and the client\_discount field
  3. Next there must be one line for each work record. This includes (in order) the work\_date, work\_description, bill\_hours. This must be sorted by the work\_date in ascending order.
  4. Make sure the same XSL file also works on the invoice2.xml file by assigning it to that file.

The final result can look something like this. Remember, there are marks for design given.



**To submit**

When you have completed the assignment, submit a zip file with all the files from the document folder to the Open Class page for this course. Remember: You MUST include a file called readme.txt that states what tool you used to complete the project.