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

# *Lab 05 – XML Schemas*

Date assigned: Wednesday, October 4, 2017

Date due: **Wednesday, October 4, 2017, 6:00 p.m.**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will have:

* Work with XML Schemas

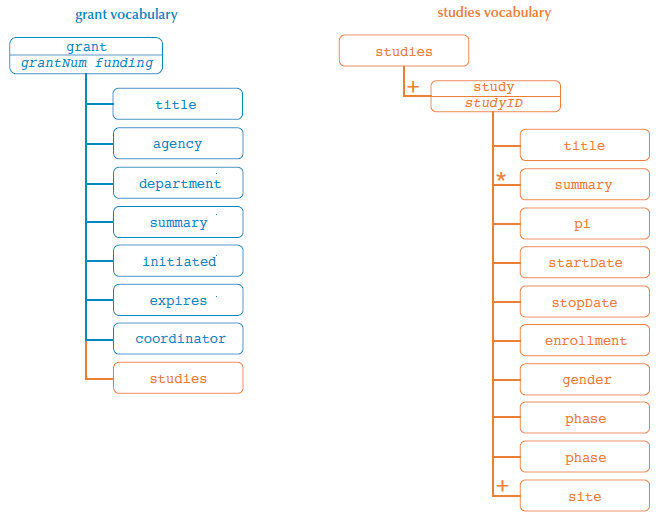
Lab Setup

1. Download the zip file which contains the source files you will need for the lab.
2. You can use any editor you choose; however, I recommend XML Spy. I would recommend you NOT use Dreamweaver.

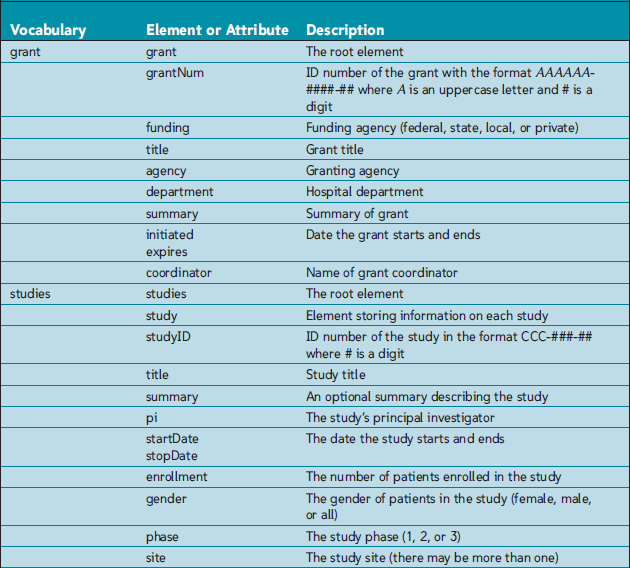
To do:

**Part A – Extending the Patient Schema**

This first problem is an extension of the patients file we worked with in class. In this exercise we will develop a schema about different grants and studies that can be done in medical research. The structure of the two vocabularies is picture below:



The following table shows the elements and attributes used in both vocabularies:



The grant.xml file contains information about a grant and the studies.xml file contains a list of studies funded by that grant. You need to create schemas to validate both the XML vocabularies. After completing that, you will create and validate a compound document containing both grant and study information.

1. Create a new schema file called grant.xsd. Add the root schema element to the document and declare the XML Schema namespace using the xs prefix (unless it was done automatically in the editor you’re using). Set the default namespace of the document to the URI http://uhosp.edu/grant/ns and make this URI the target of the schema.
2. Create the following custom data types:
   1. grantType, based on the ID data type and following the regular expression pattern “[A-Z]{6}-\d{4}-\d{2}”
   2. fundingType based on the string data type and limited to the following values: federal, state, local, and private
3. Using a Russian doll design, declare the grant element with the following nested child elements:
   1. title,
   2. agency,
   3. department,
   4. summary,
   5. initiated,
   6. expires, and
   7. coordinator.

All of the child elements should contain string data except the initiated and expires elements, which contain dates.

The grant element should also support two attributes:

1. The grantNum attribute contains grantType data; and,
2. The funding attribute contains fundingType data.
3. Open the grant.xml document. Within the root element, declare the XML Schema instance namespace using the xsi namespace prefix. Declare the grant namespace using the URI http://uhosp.edu/grant/ns and the prefix gr. Set the location of the schema to the grant namespace and the grant.xsd file.
4. Change the name of the grant element to a qualified name by adding the gr prefix to the opening and closing tags. Validate the document using XML-Spy or another parser. Correct any validation errors there may be in the data file or the schema file.
5. Create a new schema file called slist.xsd. Insert the root schema element and declare the XML Schema namespace using the xs prefix (unless it is set for you by the editor you are using). Set the default namespace and schema target to the URI http://uhosp.edu/studies/ns.
6. Create the following user-defined data types:
   1. studyIDType, based on the ID data type and following the regular expression pattern “CCC-\d{3}-\d{2}”;
   2. genderType, based on the string data type and limited to the values: female, male, and all; and,
   3. phaseType, based on the positive integer data type and having a maximum value of 3 (inclusive)
7. Apply a Venetian blind layout to this schema. Start by creating an element group named studyElements containing the following sequence of elements:
   1. title,
   2. summary,
   3. pi,
   4. startDate,
   5. stopDate,
   6. enrollment,
   7. gender,
   8. phase, and
   9. site.

The title, summary, pi, and site elements contain string data.

The startDate and stopDate elements contain dates.

The enrollment element contains a nonnegative integer.

The gender and phase elements contain genderType and phaseType data, respectively.

The summary element can occur 0 times.

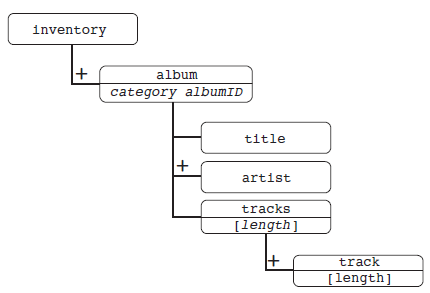
The site element must occur at least once, but its upper limit is unbounded.

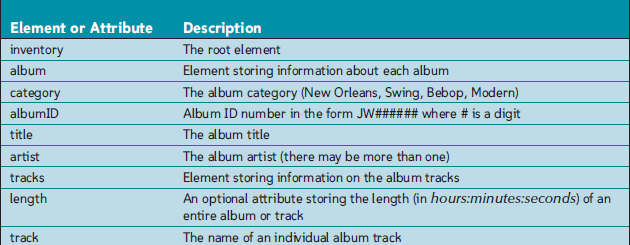
All other elements are assumed to occur only once.

1. Create a complex type named studyType. Within this complex type, insert a reference to the studyElements element group. Also declare the studyID attribute containing studyIDType data.
2. Declare the studies complex type element. Within this element, insert an element sequence containing the study element. The study element contains studyType data and must occur at least once.
3. Open the studies.xml file and declare the XML Schema instance namespace in the root element using the xsi prefix. Declare the studies namespace using the URI http://uhosp.edu/studies/ns and the prefix std. Set the location of the schema to the studies namespace and the slist.xsd file.
4. Qualify the name of the studies root element by adding the std namespace prefix. Validate the document using XML-Spy or another parser. Correct any validation errors there may be in the data file or the schema file.
5. Create a new schema for the compound document by copying grant.xsd as grantstudies.xsd. Within the root element, declare the studies namespace using the appropriate URI and the std namespace prefix.
6. Directly after the opening tag of the schema element, import the contents of the slist.xsd schema file specifying the studies namespace.
7. Directly after the declaration for the coordinator element, insert an element reference to the studies element located in the studies namespace. Save your changes to the grantstudies.xsd file.
8. Copy grant.xml to grantstudies.xml. Copy the content of the studies.xml file directly after the coordinator element. Remove the XML Schema instance namespace declaration from the studies element. Validate the contents of the grantstudies.xml file.

**Part B – Jazz**

The following diagram is an XML vocabulary structure for a music catalog. The table below is the description of the elements and attributes for this music catalog.





You are going to create a schema declaration using the Russian doll design. No namespaces need to be declared as this XML vocabulary is stand alone.

1. Create a new schema document called music.xsd and add a comment to it identifying you as the author. Insert the root schema element and declare the XML Schema namespace with an xsd prefix.
2. Define the following data types:
   1. albumIDType, based on the ID data type and following the pattern JW######, where # is a digit;
   2. jazzType, based on the string data type and limited to: New Orleans, Swing, Bebop, and Modern.
3. Declare the inventory complex element type and nest the album element within it. The album must occur at least once, but its upper limit is unbounded.
4. Within the album element, create a Russian doll layout, first nesting the child elements:
   1. title,
   2. artist, and
   3. tracks

The title and artist elements are both simple types containing string data.

The artist element may occur multiple times, but must occur at least once.

1. Add the attributes category and albumID.

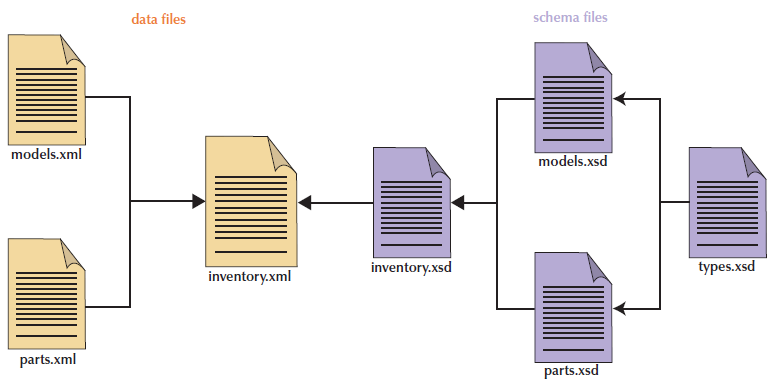
The category attribute is required and contains jazzType data.

The albumID attribute is also required and contains albumIDType data.

1. **The tracks element is a complex type element and contains at least one track element. The tracks element also contains an optional length attribute containing string data.**
2. The track element is a complex type element contains a simple text string and the length attribute. The length attribute is optional and stores string data.
3. Go to jw.xml and within the root inventory element, declare the XML Schema instance namespace. Use xsi as the namespace prefix. Attach the schema file music.xsd to this instance document. Do not place the schema or the instance document in a namespace. Validate the jw.xml file using XMLSpy or other tool. Correct any errors found.

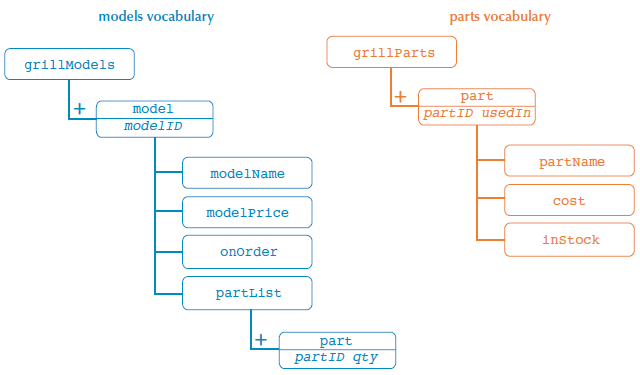
**Part C – Grills**

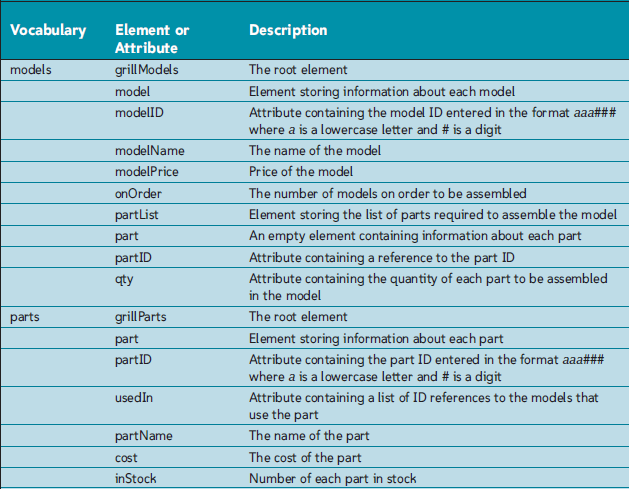
James Castillo manages the inventory for GrillRite Grills, one of the leading manufacturers of grills in North America. He has been using XML to keep records of models to be assembled and the parts required to assemble them at the Grill-Rite Grills warehouse. James needs your help in developing a system to store valid inventory data. He envisions a collection of XML documents and schema files linked in the structure displayed below:



In his proposed structure, James wants to be able to insert model and parts information in separate documents which can then be read into an inventory report through the use of external entities. Validation will be done through a collection of schemas. One schema contains a library of customized data types that James uses for his documents. The models and parts schemas validate the models and parts data. Finally, the inventory schema accesses both the models and parts schemas to validate the inventory report.

James has already laid out the structure for the models and parts vocabularies. The structure of those two vocabularies is shown below along with a subsequent table describing each element.





You are going to finish the XML document files and insert linkages between the files to create a finished product.

1. Create types.xsd and insert the root schema element, declaring the XML Schema namespace with the xs prefix. The URI of the default namespace and the schema target is http://grillrite.com/datatypes.
2. Create the following custom data types:
   1. The itemID data type, based on the ID type and following the pattern aaa###, where a is a lowercase letter and # is a digit;
   2. The itemIDREF data type, based on the IDREF type and also following the pattern aaa###
   3. The priceType data type, based on the string type and following the regular expression pattern “[$]\d+(\.\d{2})?”
3. Go to the parts.xml file. Declare the parts namespace in the grill-Parts element. The parts namespace URI is http://grillrite.com/parts and the namespace prefix is part. Qualify the opening and closing tags of the grillParts element.
4. Create a parts.xsd file and insert the root schema element, declaring the XML Schema namespace with the xs prefix. Declare the datatypes namespace using the lib prefix. Declare the parts namespace as the default namespace and make it the target of the schema.
5. Import the types.xsd file into the schema.
6. Declare the complex type element grillParts containing the part element. The part element should occur at least once in the instance document.
7. The part element contains the following child elements:
   1. partName,
   2. cost, and
   3. inStock;

and the attributes:

* 1. partID and
  2. usedIn.

The partName element contains string data.

The cost element contains priceType data.

The inStock element contains nonnegative integers.

The partID attribute contains itemID data and the usedIn attribute contains IDREFS data.

1. Go to the models.xml file. Add a namespace declaration to the root grillModels element using the URI http://grillrite.com/models and the prefix mod. Qualify the opening and closing tags of the grillModels element.
2. Create a models.xsd and insert the schema element to the file, once again declaring the XML Schema namespace using the xs prefix. Declare the datatypes namespace using the lib prefix. Set the models namespace as the default namespace and target of the schema.
3. Import the types.xsd file into the schema.
4. Using a Russian doll design, declare the complex type element grillModels.
   1. Nested within this element, declare the model element, allowing it to occur at least once.
   2. The model element contains the child elements
      1. modelName,
      2. modelPrice,
      3. onOrder, and
      4. partList,
      5. and the modelID attribute.

The modelName element contains string data.

The modelPrice element contains priceType data.

The onOrder element contains nonnegative integers.

The modelID attribute contains itemID data.

The partList element is a complex type element containing one or more part elements. Each part element is an empty element containing the partID and qty attributes. The partID attribute stores itemIDREF data. The qty attribute stores positive integers.

1. Create inventory.xsd. Declare the XML Schema namespace in the root schema element using the xs prefix. Declare the models and parts namespaces using the namespace prefixes mod and parts respectively. Use the URI http://grillrite.com/inventory as the default namespace and target of the schema. Import the models.xsd and parts.xsd schemas.
2. Declare a complex type element named inventory containing the element references grillModels from the models namespace, and grillParts from the parts namespace.
3. Open the (nearly empty) inventory.xml file. Copy and paste the content from the models.xml document within the inventory element. Copy and paste the content from the parts.xml document within the inventory element directly below the models data.
4. Within the inventory element, declare the XML Schema instance namespace. Declare the inventory namespace using inv as the namespace prefix. Attach the inventory.xsd schema. Qualify the opening and closing tags of the inventory element.
5. Validate the inventory.xml file. Note that the validation errors will be discovered in the models.xml and parts.xml files. Correct any mistakes you find in those files.

**To submit**

When you have completed the lab exercise, create a zip file of the folder and load it to the Moodle page for this course.