

Pre-Lab XML Activity: An Introduction to XML

Must be completed by Sunday September 13 at 11:00pm for students enrolled in Monday lab sessions
Must be completed by Tuesday September 15 at 11:00pm by students enrolled in Wednesday lab sessions
Must be completed by Thursday September 17 at 11:00pm by students enrolled in Friday lab sessions

FOLDER/FILE SUBMISSION TO THE APPROPRIATE KSU ONLINE DROPBOX MUST BE COMPLETED BY THE DATE AND TIME LISTED ABOVE.

If you meet the folder/file submission deadline listed above, **you can hand in your packet of printouts at your regularly scheduled lab session the next day.**



Before you start, PLEASE print out this entire tutorial. Having a printed copy of the tutorial in front of you results in the activity taking less time than completing the activity while constantly switching back and forth between the instructions and the application you are working with. You will get finished with this activity quite a bit faster if you have printed activity description in front of you.

XML (eXtensible Markup Language) is an emerging technology for creating web documents that actually describe the data being transmitted. XML will form the basis of future EDI (electronic data interchange) and electronic commerce. While there are lots of books and tutorials available regarding XML, most simply focus on the technical aspects of this technology. This tutorial focuses on what you (the business student) need, which is business applications of XML and XML technologies.


Note: The information in this document is synthesized from materials developed by Skip White – the University of Delaware, Dale L. Lunsford – University of Alabama, Pascal Bizarro – University of Mississippi, Brian Kovar – Kansas State University, and Stacy Kovar – Kansas State University. Brian Kovar has made presentations over this document's contents at the 2004 Accounting Information Systems Educator Conference and at the 2004 American Accounting Association Midwest section meeting. Some of the descriptions provided also come from Information Systems: Creating Business Value, authored by Huber, Piercy and McKeown (published by Wiley).

The XML Tutorial is based upon some of the work done by the individuals listed above. They have reported that responses of their students to the tutorials and exercises that our tutorial is based upon have been very positive, with students commenting that they now understand the importance of XML. Students have also reported that they have become involved in extensive conversations with prospective employers about XML during job interviews and that the potential employers have responded positively to the students' level of knowledge. Students have also reported that they successfully used XML on-the-job during internships after completing the tutorials that our tutorial is based upon.

After completing our tutorial, you can add a statement to your resume that you "have a basic understanding of XML and XML technologies." In order to complete the tutorial, you will also need to copy the files that go along with the tutorial over to the storage media that you use for this class. Those files are located in the Information folder in S drive.

-  Prior to starting this activity, you need to download the activity files from KSU Online. After logging in, visit **Files & Content**, then **Lab Assignments** and then **XML Pre-Lab files to download to your storage media**.
-  Once inside the "XML Pre-Lab files to download to your storage media folder", you will see several files that you need to download into the Internet Project folder that you created in a prior pre-lab activity (recall you were told "**ALL OF THE FILES THAT ARE PART OF YOUR INTERNET PROJECT NEED TO BE STORED IN THIS FOLDER.**" This includes all of the files that you create or download as part of this tutorial, all of the **XML files** used in a future pre-lab exercise, and the business site that you create in a future activity as well. If you

don't have **everything** in this folder, future problems will result (and who wants that)."

 To download those XML files from KSU Online, simply **right-click** each file and select either the **SAVE TARGET AS** or **SAVE LINK AS** option (depending on the browser you are using and specify that you want to store the files in the **Internet Project folder** that you created in a prior pre-lab activity. The four files you need to download are:

- Wellformed.xml
- Notwellformed.xml
- Games.xml
- My XML tags.docx

As you work through the tutorial, do the activities and exercises that you encounter. **PLEASE READ EVERYTHING CAREFULLY.**

An Introduction to XML

You see the word **Manhattan**. Exactly what is **Manhattan**? In answering this question you would say, "That is easy. It is the city where I go to college." However, to a computer software program, **Manhattan** is just a string of characters. The computer can read the characters, but the computer does not know what **Manhattan** is.

Markup languages are used to provide additional information to a computer about a string of characters.
<city>Manhattan</city>

HTML (HyperText Markup Language) is one type of markup language that you may have already encountered. HTML is the standard markup language that is used to give the computer information about how to display characters on web pages. <h1>Manhattan</h1> tells a web browser to display Manhattan in the largest font size possible. HTML is quite useful for displaying information on web pages, but there still must be a person interpreting what data is contained on the web page. As far as the computer is concerned, the string **Manhattan** could be a drink that I order at a bar, the name of a country, the name of a state, the name of a city, the name of particular movie, the name of a clothing item, the possibilities are endless. This limitation has led to the development of another markup language, XML.

XML (Extensible Markup Language) is similar to HTML in that it uses tags. However, whereas HTML tags are used by the browser to *display* content correctly, the tags in XML are used to provide *information about the content itself*. In this way, computer programs can be created to find and manipulate specific content, not just display it. XML tags are much more powerful than HTML tags and they can be customized for any kind of information, as long as both the sender and the receiver of the XML document agree on the definitions of the tags. The "extensible" portion of the name "Extensible Markup Language" means that new tags can be created at any time, extending the language to fit any kind of information.

A good way to understand XML is to compare it to HTML, the markup language used to create web pages. HTML is a formatting language that is meant to display numbers and text in a predefined way when viewed using a web browser. As such, it does not have the structure to impart meaning to items like part numbers or prices. Because it is not a good idea to have computers trying to infer meanings from entries on web pages, HTML is not appropriate for transmitting large amounts of purchasing and shipping information over the Internet. HTML's predetermined tags also reduces its flexibility and the only way to retrieve information from a web page is to search for specific text, assuming you even know what text is going to appear on the web page.

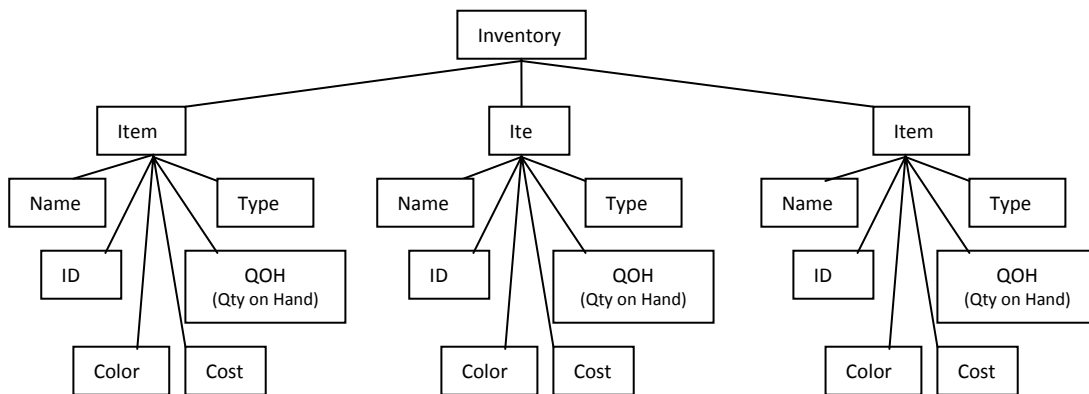
XML solves these HTML problems because it is a language that emphasizes the **structure** and **meaning of data**. This means that product information such as name, identifier, price, etc. are easily transmitted using XML. XML is also flexible in that users can define their own tags. For example, a company can use XML to define tags that their trading partners can understand. This also enables searches over the data using the meaning of the data instead of just using a text search. For example, the tag <PARTID> would indicate that the field that followed was a part

number and it would be easy to find all part numbers by searching for this tag. An XML file can be processed purely as data by a program, it can be stored with similar data on another computer or, like an HTML file, it can be displayed

The core XML Technologies are: the XML Schema, the XML Document, and XSL (the extensible Stylesheet Language).

XML Schema: An XML schema provides a standard data structure that allows organizations and their different computer programs to recognize and share data. An XML schema describes the common elements of a specific type of document (for example a recipe, a financial statement or a bank account) and correspondingly the names of the tags that can be used to describe data in those documents. It also describes the type of data that is acceptable for a particular element (i.e. either a list of acceptable values or a restriction that the values must be numeric or text). Therefore, schemas must be fairly complex and comprehensive. Schemas allow a hierarchical structure for data, which can be depicted as an inverted tree, as shown below.

The tree-like structure that follows **represents a schema**, but the diagram itself **is not a schema** (just its representation). For instance, let's pretend that a business want to both store and transmit information related to its inventory. The resulting XML document will have a root element, which describes the content of the entire document (in this case: Inventory). A business's inventory listing can contain many Item elements. Each item element consists of six attributes (descriptors): Name, ID, Color, Cost, QOH (quantity on hand) and Type. In this example, only three inventory items are being shown, but of course, there could be many more. For each item in inventory, you would branch it off the root element, so there could be hundreds or even thousands of branches off the root (Inventory, in this case), and each branch/item would be described by the same six attributes.



XML schemas are created using a formal language called the XML Schema Definition (XSD) language. Two sample XML schemas follow this paragraph. The schema on the left specifies the tags and characteristics of the inventory example seen earlier. The schema on the right is a simple XML schema for a bank account. Notice that the XML schema defines the tags, the organization of the elements, and the data type for each element.

```

<?xml version="1.0"?>
<!-- schema for Bobsports.com inventory -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="inventory">
    <xsd:complexType>
      <xsd:element name="item" minOccurs="1" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:element name="name" minOccurs="1" maxOccurs="1"/>
          <xsd:element name="id" minOccurs="1" maxOccurs="1"/>
          <xsd:element name="color" minOccurs="1" maxOccurs="1"/>
          <xsd:element name="weight" minOccurs="1" maxOccurs="1">
            <xsd:attribute name="units" use="required"/>
          </xsd:element>
          <xsd:element name="cost" minOccurs="1" maxOccurs="1">
            <xsd:attribute name="units" use="required"/>
          </xsd:element>
          <xsd:element name="qoh" minOccurs="1" maxOccurs="1"/>
          <xsd:element name="type" minOccurs="1" maxOccurs="1"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="BankAccount">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="AccountID" type="xsd:string"/>
        <xsd:element ref="AccountHolders"/>
        <xsd:element name="Balance" type="xsd:decimal"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="AccountHolders">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="AccountHolder" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="AccountHolder">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="HolderName" type="xsd:string"/>
        <xsd:element name="HolderTaxID" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

In most cases, the XML schema organizes the elements as they should appear in the XML document. Basically, an XML schema performs a function similar to a data dictionary in a database management system. XML schemas are being developed for a number of common business applications. Accounting firms and government agencies are working on developing schemas called XBRL, which will be used to simplify the distribution of financial reports. XBRL (eXtensible Business Reporting Language) is a specific XML schema that describes the elements/data in financial statements. It is an XML-based specification for publishing financial information. XBRL makes it easier for public and private companies to share information with each other, industry analysts, and with shareholders. XBRL includes tags for data such as annual and quarterly reports, SEC filings, general ledger information, net revenue and accounting schedules. Other schemas are also being developed to facilitate the transfer of financial information between banks, businesses, and customers. Other schemas are being developed to transfer business transaction information through the Internet in an easy and low-cost manner so as to simplify and expand electronic business opportunities worldwide. The Internal Revenue Service is actively engaged in a number of initiatives to utilize XML for tax reporting and electronic filing.

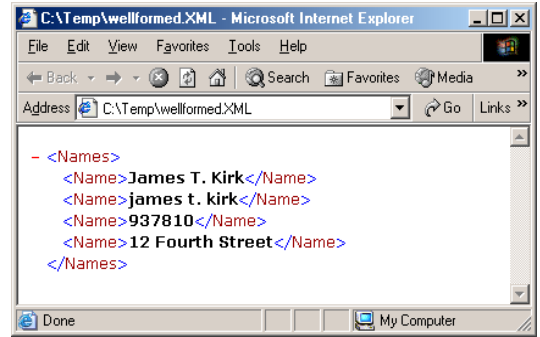
As a business student who might encounter XML technologies in the workplace, it is very important to know what a schema is and how it is used. The following paragraph does a good job of summarizing what a schema is and how it is used.

XML is directly usable over the Internet, and many companies that are engaged in electronic commerce are exchanging XML transaction data. When engaging in e-commerce, a seller's computer can receive an order generated by the buyer's computer and immediately understand the information contained in the order. An XML schema makes this possible by describing the rules that "valid" XML documents must follow. Applications that understand a schema know what to expect from other applications/computers that create documents that follow that schema. They can process valid documents because they understand the content, organization, and structure of XML instance documents that abide by that schema. In addition to describing their structure, schemas can be used to validate XML instance documents. If you are an e-business and you want to share and automatically process XML documents with your business partners, then you need schemas to define the rules that your instance documents will follow and the rules that your partners' documents follow; for example, what constitutes a valid entry in a field like payment terms or shipping method. Then, you share your schemas with your business partners so that all of you can accurately process each others' XML documents.

XML Document: The XML document is the heart of XML. Each XML document stores data consistent with an XML schema. If the XML schema is the data dictionary, the XML document is the database, stored with tags around each piece of data. The tags contain the field or element name from the XML schema.

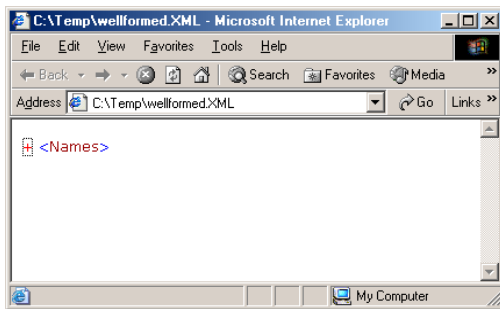
Technically, an XML document can exist without a corresponding XML schema, as long as it is well-formed. This simply means each element starts with an opening tag and ends with a closing tag, and that the opening tags and

Well-formed XML Data: Open Internet Explorer. In Internet Explorer, open *WellFormed.xml* (click *File* → *Open* → *Browse*, navigate to the location where you stored the XML files, change *Files of type* to *All Files*, select *WellFormed.xml*, click *Open*, and click *Ok*). Internet Explorer displays the well-formed file, as shown.



Each XML document must have a root element describing the content of the document. Because of this, *WellFormed.xml* has a *Names* root element, indicating that the document consists of a list of names; the first line shows the *<Names>* opening tag and the last line shows the *</Names>* closing tag. Notice that Internet Explorer shows all tags and data.

Click the minus sign (-) beside the *<Names>* opening tag. Internet Explorer *collapses* the selected element as shown in the next illustration. **NOTE:** If you are unable to expand and collapse the document, **you might need to turn off Internet Explorer’s content filtering** (message regarding this might be displayed in your browser above the instance document and below the URL). Click where it say “Click Here For Options,” and then “**Allow Blocked Content.**” You might need to do this for each XML document that you wish to open.



If you are unable to expand and collapse the document, **you might need to turn off Internet Explorer’s content filtering** (seen above the document and below the URL).
Click where it say “Click Here For Options,” and then “**Allow Blocked Content.**” You might need to do this for each XML document that you wish to open.

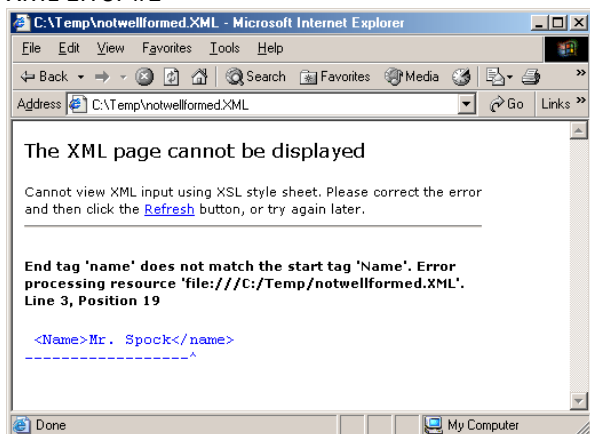
Click the plus sign (+) to *expand* the element so that the document looks like the Wellformed (Expanded View) picture seen previously.

The only other element besides the root element in the list is the *Name* element. As shown in Wellformed (Expanded View), each individual name element rests between a *<Name>* opening tag and *</Name>* closing tag. You have probably noticed that most of the names do not make much sense. Do not worry about this yet; we will address the actual contents of the elements later on.

Problems with XML Data: XML data are not well formed if there are any problems with the tags used or the tags do not balance. When an XML-enabled application, like Internet Explorer, encounters a problem with XML data, the application displays an error message.

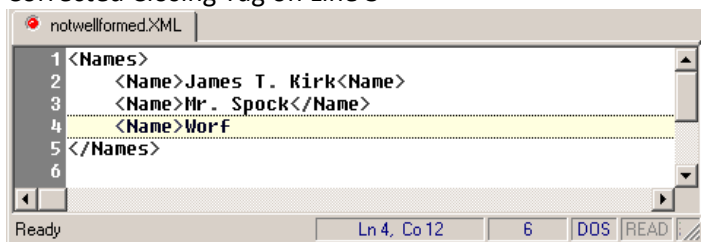
Open *NotWellFormed.xml* in Internet Explorer. Internet Explorer displays an error message similar to the picture below.

XML Error #1



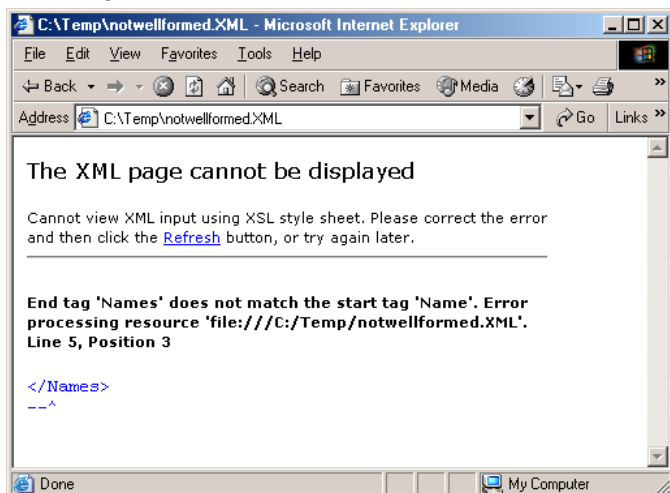
- Let's open the original XML document that was created so that we can find and fix the error.
- Leave Internet Explorer open and open *NotWellFormed.xml* in Notepad.
- On line three (Mr. Spock), the closing tag starts with a lowercase letter but the opening tag starts with an uppercase letter. This is the **first error in the file**, but there may be other errors that we will soon find and correct. Correct the case of the *closing tag* as shown below (on the Mr. Spock line). Save the file.

Corrected Closing Tag on Line 3



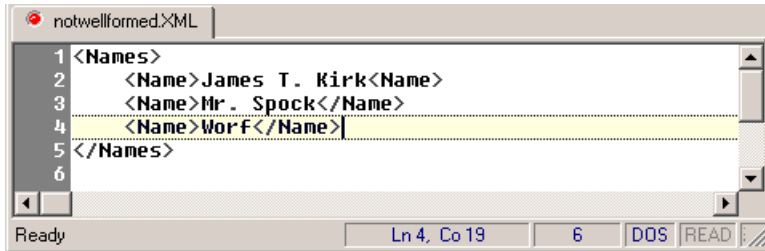
- Refresh the page in Internet Explorer. Internet Explorer detects another error and displays an error as shown next.

XML Error #2



- In this case, Internet Explorer reports that the `</Names>` closing tag and `<Name>` opening tag do not balance. To track down the source of this error, go back to the file that you have opened in Notepad. As indicated in the prior picture, start with line #5 and work backwards. Notice that the element on line 4 does not have a closing tag. In Notepad, add a closing tag on line 4 (`</Name>`) as shown below. Save the file.

Line 4 Correction

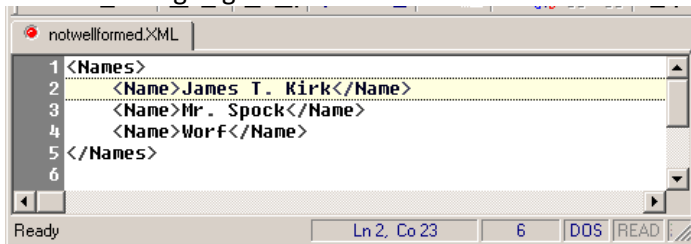


```
notwellformed.XML
1 <Names>
2   <Name>James T. Kirk<Name>
3   <Name>Mr. Spock</Name>
4   <Name>Worf</Name>
5 </Names>
6
```

Ready Ln 4, Co 19 6 DOS READ

- Refresh the page in Internet Explorer. Unfortunately, Internet Explorer reports the same error seen above in XML Error #2. This means a problem with tags balancing still exists.
- Working backwards starting at line 4, verify that each element has a valid opening tag and closing tag. Lines 3 and 4 are ok; however, the closing tag in line 2 does not have a slash. Add the slash to the closing tag (see below). Save the file.

Slash in Closing Tag

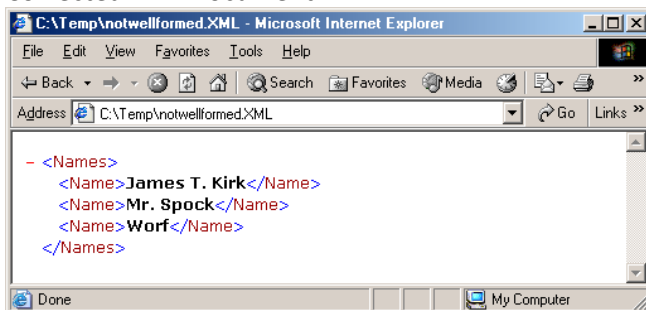


```
notwellformed.XML
1 <Names>
2   <Name>James T. Kirk</Name>
3   <Name>Mr. Spock</Name>
4   <Name>Worf</Name>
5 </Names>
6
```

Ready Ln 2, Co 23 6 DOS READ

- Refresh the page in Internet Explorer. At this point, the document is well formed, as seen in the next illustration.

Corrected XML Document



```
C:\Temp\notwellformed.XML - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Home Search Favorites Media
Address C:\Temp\notwellformed.XML Go Links
- <Names>
  <Name>James T. Kirk</Name>
  <Name>Mr. Spock</Name>
  <Name>Worf</Name>
</Names>
Done My Computer
```

The problems encountered illustrate several rules related to XML documents:

1. XML is case sensitive – uppercase and lowercase characters are different.
2. Opening and closing tags must be the same, except the closing tag starts with a slash (/).
3. Tags must balance – every opening tag must have a closing tag.

One additional rule: XML tags **cannot contain spaces separating words**. `<First Name>` is not a valid XML tag. `<FirstName>` and `<First_Name>` are valid XML tags.

- Print out **notwellformed.xml** in Internet Explorer (since you have now made it well-formed). Label the printout (by hand) as **notwellformed.xml**

Exercise #1: Finding and Detecting XML Errors

Earlier in this tutorial, you had a chance to see the rules that make an instance document well-formed. In this exercise, you will now apply those rules.


- Open the file called **games.xml** in both Notepad and Internet Explorer. The instance document contains a number of coding errors (8 errors by my count, but some people might say there are nine errors). Please locate and fix each of the errors so that **games.xml** becomes a well-formed XML instance document. Once the document is well-formed:
 - Make a printout of your XML code, as seen in Notepad.
 - Make a printout of your well-formed XML instance document, as seen in Internet Explorer.
 - Label both of your printouts as Exercise #1.

Exercise #2 – Determining What Your XML Tags Will Be

Up to this point, you have been reading about XML and working with XML documents created by someone else. Now, it is time to plan, and then eventually make, your own XML instance document.

In the upcoming **Internet Project** that will be due in the near future, you are going to create a business web site so you are going to have to decide what type of product you wish to sell, find or create pictures related to that product, and also come up with product descriptions. (The web page that you made as part of the HTML pre-lab activity will also be part of your site as well).

- Right now, if you haven't done so already, you now need to decide on what product you want to sell on your web site.
- As part of the Internet Project, you will also need to provide links to XML documents where customers can go to find out more information related to your offerings. The basic building block of the XML portion of the assignment is the XML instance document, and before you can make one of those, you need to determine what your XML tags will be.
- Open the word processing file called **My XML tags.docx**
- That file is used to plan the XML instance document that you will create as part of your upcoming project. Keeping in mind the business web site that you are going to create, read the text found in the document and replace the words appearing in red font color with one word NOUNS appropriate to what you plan to offer on your business web site. These words that you will be adding need to be **NOUNS** that are **one word only** since XML tags are typically one word and you can't have spaces in your XML tags. Each of the words that you add to the word processing document needs to be **generic/non-specific wording**. **Real data (specific wording) will be added in the future** (but now is not the time)
- Go ahead and read through the word processing document, making the appropriate changes/substitutions.
- Once you have finished, save your work. Make two printouts of the word processing file. Include one printout in the packet of printouts that you hand in as part of the pre-lab activity. **Bring the other printout to lab since you will be using it and possibly modifying it during your lab session.**

 Once you have made the required printouts, all you have left to do is submit your work for grading.

Please make sure that you keep the files that you created in this activity since they will also be part of the Internet Project that you will be submitting in the future.

HANDING IN THE ACTIVITY

1) When you are ready to turn in your work, make sure that you place or have **ALL OF YOUR ACTIVITY FILES IN A NAMED FOLDER/DIRECTORY**. Create a zip/compressed file, and then submit the zipped/compressed file to the appropriate KSU Online Dropbox. In case you have forgotten how to submit files to KSU Online, please visit <http://info.cba.ksu.edu/bkovar/AssignmentSubmission.htm>

2) Please assemble your packet of printouts. The printouts should be arranged in the following order:

- A cover page containing the required information (outlined in the class syllabus).
- Print out notwellformed.xml in Internet Explorer (after you have made it well-formed). Label the printout as **notwellformed.xml**
- Make the printouts specified in Exercise #1. The printouts should already be labeled.
- Hand in the word processing document created Exercise #2.

IN ORDER TO BE CONSIDERED FOR FULL CREDIT, YOUR PRINTOUTS MUST BE IN THE REQUIRED SUBMISSION ORDER. Deductions will be taken for printouts that are not in the proper order.

FOLDER/FILE SUBMISSION MUST BE COMPLETED BY THE DUE DATE/TIME LISTED ABOVE. If you meet the folder/file submission deadline listed above, you can hand in your packet of printouts during your regularly scheduled lab session the next day.