Lab Assignments

Outline:

1. XML

   1.1. XML Syntax & DTDs
      1.1.1. Introduction to XML Syntax & DTDs
      1.1.2. Advanced XML Syntax & DTDs

   1.2. XML Schema
      1.2.1. Introduction to XML Schema
      1.2.2. Advanced XML Schema
      1.2.3. (optional: extra development exercise 1)

   1.3. XPath
      1.3.1. Introduction to XPath
      1.3.2. Advanced XPath Queries

   1.4. XSLT
      1.4.1. Introduction to XSLT
      1.4.2. Advanced XSLT
      1.4.3. Presenting the XML

Lab 1.1.1
Introduction to XML Syntax & DTDs

Required files: amazon.xml

Learning outcomes:
1. Pay attention to specific XML Syntax peculiarities.
2. Learn how to construct a valid DTD and associate it with an XML document.
3. Get acquainted with the EditX environment.

1. Open file "`amazon.xml`" with EditX.

The file describes a small subset of the books available at www.amazon.com. The description for each book includes:

- the category the book belongs to,
- the book title,
- the list of authors,
- the user rating for the specific book,
- the book price.

2. Check if the XML document is well-formed (i.e. syntactically correct). Correct potential errors.

Instead of manually inspecting the whole document, you are advised to use EditX's "*Check Document*" functionality.
Tip: There are 5 syntactic errors in the document.

3. Construct a DTD document that correctly describes the XML document "`amazon.xml`". Declare the DTD association inside the XML document.

All the above tasks should be completed inside the EditX environment.
Pay specific attention, when declaring the occurrence of each element.
Before associating the DTD with the XML, make sure that the DTD document is error-free.
Instead of manually associating the DTD with the XML, you are advised to use EditX's "*Assign DTD to document*" functionality.
Tip: Practical and brief DTD tutorial: http://www.w3schools.com/dtd/default.asp.

4. Extend the DTD to include further types of items besides books, like software.

You can add actual item types featured in http://www.amazon.com, or you can create your own custom types. E.g.: video games, apparel, household items etc.

5. Extend the XML with examples of the items added in the previous step.

Include at least two sample items for each type you created during the previous task.

Lab 1.1.2
Advanced XML Syntax & DTDs

Learning outcomes:
1. Deeper DTD & XML Syntax understanding.
2. Modeling a domain of interest in DTD and XML.
3. More experience with the EditX environment.

1. Choose a domain of interest (e.g. e-shop, library, recipes) and determine its basic elements (e.g. a library has books, employees etc.) as well as their respective attributes (e.g. a book has a title, a list of authors etc.). Draw a small diagram containing the elements, their attributes, as well as the interrelationships among the elements.

Include in your model at least 3 but no more than 7 elements.
There is no limit in the number of attributes, but avoid exhaustive descriptions.

2. Use EditX to construct a DTD document that describes the domain modeled during the previous step.

Tip: Practical and brief DTD tutorial: http://www.w3schools.com/dtd/default.asp.
Can any attribute be declared as an element and vice-versa? Which modeling path do you eventually choose and why? (*Tip*: See the following for ideas: http://www.w3schools.com/dtd/dtd_el_vs_attr.asp)
Pay specific attention, when declaring the occurrence of each element.
Try to include ID and IDREF(S) attributes as well as REQUIRED attributes in the DTD.
Make sure that the final DTD document developed is error-free.
Tip: See the following sample DTDs for further ideas regarding the modeling of the domain: http://www.w3schools.com/dtd/dtd_examples.asp

3. Use EditX to construct an XML document that conforms to the DTD developed during the previous step. Declare the DTD association inside the XML document.

Don't create more than 2 sample instances for each element in the DTD.
Instead of manually associating the DTD with the XML, you are advised to use EditX's "*Assign DTD to document*" functionality.

4. Validate the XML document developed during the previous step. Correct potential errors.

Tip: You can use EditX for validation, or you can also try an on-line validator, like: http://www.xmlvalidation.com/

Lab 1.2.1
Introduction to XML Schema

Required files: amazon.xsd, amazon.xml, amazon.dtd

Learning outcomes:
1. Pay attention to specific XML Syntax peculiarities.
2. Learn how to construct a valid XML Schema and associate it with an XML document.
3. Learn why XML Schemas are more powerful than DTDs.

1. Open files "`amazon.xml`", "`amazon.dtd`" and "`amazon.xsd`" with EditX.

The "amazon.xsd" is an XML Schema document that describes part of the structure of the "amazon.xml" XML document presented in Lab 1.1.1. Actually, XML Schema is a richer and more powerful XML-based alternative to DTD and its purpose is to define the legal building blocks of an XML document, just like a DTD (here "**`amazon.dtd`**" created also in Lab 1.1.1). Mention that XML documents can have a reference to a DTD or to an XML Schema.
Look at these documents and briefly compare DTD with XML Schema. Pay specific attention to data facets (restrictions on data) and data patterns (data formats).
Tip: See http://www.w3schools.com/Schema/schema_howto.asp for a relevant example.

2. Complete the XML Schema document by adding the missing information.

Tip: A mixed complex type describing the "book" element is missing.
(Don't forget to include the "category" attribute!)
Mixed complex type elements can contain attributes, elements, and text; study them in http://www.w3schools.com/schema/schema_complex_mixed.asp

3. Modify the XML document ("`amazon.xml`") by declaring the XML Schema association inside the XML document.

You have to alter the existing reference (to the DTD document) with a new one.
Tip: See http://www.w3schools.com/schema/schema_howto.asp for information about references (both to DTD and XML Schema).

4. Validate the XML Schema document completed previously. Correct potential errors.

Tip: You can use EditX for validation, or you can also try the official W3C validator for XML Schema: http://www.w3.org/2001/03/webdata/xsv.

Lab 1.2.2
Advanced to XML Schema

Required files: (optional) Xml document developed in Lab 1.1.2
Learning outcomes:
1. Deeper XML Schema understanding.
2. Modeling a domain of interest in XML Schema.
3. More experience with the EditX environment.

1. Choose a domain of interest (e.g. e-shop, library, recipes) and develop a valid XML document or use the one developed in Lab 1.1.2.

Tip: If you choose to develop a new XML document repeat steps 1 and 4 (Lab 1.1.2). Otherwise, take a look at the existing XML document. Pay attention to its elements, its attributes and the interrelationships among the elements.

2. Use EditX to construct an XML Schema document that describes the domain chosen during the previous step.

To create the schema you could simply follow the structure in the XML document and define each element as you find it. This method is quite simple and naïve but it is important for understanding the XML Schema.
Tip: See the following XML Schema sample (*Create an XML Schema*) for further ideas: http://www.w3schools.com/schema/schema_example.asp.
Take into account simple and complex element types and define the number of possible occurrences for an element with the maxOccurs and minOccurs attributes.
Finally, don't forget to start with the standard XML declaration followed by the xs:schema element that defines a schema:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
</xs:schema>
```

3. Modify the initial XML document by declaring the XML Schema association inside the XML document.

You have to alter the existing reference to the DTD document (if there is one) with a new one to the XML Schema.
Tip: See http://www.w3schools.com/schema/schema_howto.asp and/or exercise 3 Lab 1.2.1.

4. Validate the XML Schema document completed previously. Correct potential errors.

Tip: You can use EditX for validation, or you can also try the official W3C validator for XML Schema: http://www.w3.org/2001/03/webdata/xsv.

5. Use EditX to construct an advanced XML Schema document that describes the domain chosen previously.

The previous design method is very simple, but can be difficult to read and maintain when documents are complex. An alternative design method is based on defining all elements and attributes first, and then referring to them using the ref attribute.

Tip: See the following XML Schema sample (*Divide the Schema*) for further ideas: http://www.w3schools.com/schema/schema_example.asp.

6. Modify the initial XML document by declaring the XML Schema association inside the XML document.

Once again you have to alter the existing reference to the DTD document with a new one to this XML Schema.

Tip: See http://www.w3schools.com/schema/schema_howto.asp and/or exercise 3 Lab 1.2.1/Lab 1.2.2.

7. Validate the XML Schema document completed previously. Correct potential errors.

Tip: Once again you can use EditX for validation, or you can try the official W3C validator for XML Schema: http://www.w3.org/2001/03/webdata/xsv.

Lab 1.3.1
Introduction to XPath

Required files: books.xml
Learning outcomes: 1. Understanding of XPath expression execution and results.
2. Familiarization with EditX's XPath Panel.

1. Open the supplied XML file "`books.xml`" and execute the following XPath expressions in EditX. What is the result of each path expression?

- `catalog/book`
- `//author`
- `//book/title`
- `//book/@id`

The aim is to get familiarized with EditX's XPath view.

2. Execute the following XPath expressions in EditX. What is the result of each path expression?

- `catalog/book[2]`
- `catalog/book[last()]`
- `catalog/book[last()-2]`
- `catalog/book[position()<3]`
- `catalog/book[price>10]`
- `catalog/book[price>10]/title`
- `catalog/book[@id]`
- `catalog/book[@id="bk102"]`

Expressions like these that contain predicates are used in retrieving specific nodes or nodes that contain a specific value.
Predicates are always embedded in square brackets.

3. Execute the following XPath expressions in EditX and examine the results.

- `catalog/*`
- `//*`
- `//book[@*]`
- `//book/node()`
- `//book[2]/node()`
- `//book/title | //book/price`

XPath wildcards can be used to select unknown XML elements.
The "|" operator in an XPath expression retrieves several paths.

Lab 1.3.2
Advanced XPath Queries

Required files: books.xml
Learning outcomes: 1. Deeper understanding of XPath advanced queries.
2. Familiarization with EditX's XPath Panel.

1. Open the supplied XML file "`books.xml`" and execute the following XPath expressions in EditX. What is the result of each path expression?

- `child::book`
- `attribute:: id`
- `child::*`
- `attribute::*`
- `child::text()`
- `child::node()`
- `descendant::book`
- `ancestor::book`
- `ancestor-or-self::book`
- `child::*/child::price`

All of the above expressions require specifying a current node.
Execute the expressions each time from the "current node" (designated by the corresponding selection button in EditX).
Pay specific attention when choosing the current node.

2. Execute the following XPath expressions in EditX. What is the result of each path expression?

- `//book | //author`
- `catalog/book[price!=5.95]`
- `catalog/book[price>=4.95]`
- `catalog/book[price=4.95 or price=5.95]`
- `catalog/book[price>=4.95 and price<10]`
- `count(//book[price>10])`
- `//*[name()='title']`
- `//*[starts-with(name(),'p')]`
- `//*[contains(name(),'p')]`
- `//*[string-length(name()) >6]`

Lab 1.4.1
Introduction to XSLT

Required files: amazon-extended.xml, amazon.dtd

Learning outcomes:
1. Understanding of basic XSLT transformations and XSLT+XML integration.
2. Familiarization with executing XSLT transformations via EditX.

1. **Open the supplied XML file "`amazon-extended.xml`" (or any other similar XML file you may have created during previous lab assignments). The file contains 2 types of products: books and software. Create an XSL transformation document for presenting all the products in the XML file. Transform the XML using the XSLT you developed. Save the developed XSL transformation file as: "`all-products.xsl`". View the resulting HTML file in a browser window.**

The new XSLT document should be created from within EditX, by using the command "*File* > *New* > *XSL Transformation for HTML output*"
You should apply EditX's "*Transform using XSLT*" command (from within "*XSLT/XQuery*" menu), designating the output HTML.

2. **Associate the XML file "`amazon-extended.xml`" (or any other similar XML file you may have created during previous lab assignments) with the "`all-products.xsl`" you created previously. Try to open the XML file inside a browser window.**

If the browser is XML- and XSLT-compliant, it will apply the XSL transformation onto the XML file and display its content accordingly (All major browsers have support for XML and XSLT).
The output should be identical to the HTML output during the previous step.
Try to change some of the data inside the XML document and refresh the browser window: the result is updated instantly!

Lab 1.4.2
Advanced XSLT

Required files: amazon-extended.xml, amazon.dtd
Learning outcomes:
1. Understanding of advanced XSLT transformations.
2. Familiarization with executing XSLT transformations via EditX.

1. Update the "`all-products.xsl`" you created previously, by adding sorting capabilities. Sort the products by product title and by product price. Save the updates as two new XSLT files: "`all-products-title.xsl`" and "`all-products-price.xsl`".

Notice what happens when applying the "**`<xsl:sort>`**" element in numeric values, like prices.
Tip: If we want XML elements to be treated as actual numeric values by XSLT, then a simple addition in the "**`<xsl:sort>`**" instruction is needed:

```
<xsl:sort  select="price"  data-type="number"/>
```

2. Open the supplied XML file "`amazon-extended.xml`" (or any other similar XML file you may have created during previous lab assignments). The file contains 2 types of products: books and software. Create an XSL transformation document for:

- **Sorting all the products by price in descending order.**
- **Sorting all the products first by price and then by title (two sorting keys in this order).**
- **Sorting all the products by category.**
- **Displaying all products with price > 15.**
- **Displaying all products with price > 15 in red font and all the rest in green.**
- **Displaying all products with price > 25 in red font, all products with price > 15 in orange font and all the rest in green.**

All XSLT documents should be created from within EditX.
You should apply EditX's "*Transform using XSLT*" command (from within "*XSLT/XQuery*" menu), designating the output HTML.

Lab 1.4.3
Presenting the XML

Required files: all-products-title.xsl, all-products-price.xsl, all-products.xsl, amazon-extended.xml

Learning outcomes: Learn how to have the source data stored in a single location and present its various views in different webpages.

1. Make sure you have 3 XSLT files (developed during previous labs): One that displays all the products ("`all-products.xsl`"), one that displays the products sorted by title name ("`all-products-title.xsl`") and one that presents them sorted by price ("`all-products-price.xsl`"). Create 3 new XML files with the same names that comply to the following:

- **`all-products.xml` applies `all-products.xsl`.**
- **`all-products-title.xml` applies `all-products-title.xsl`.**
- **`all-products-price.xml` applies `all-products-price.xsl`.**

Inside each of the 3 XML files include the initial XML ("`amazon-extended.xml`") as ENTITY and create an "`amazon`" root element.

There are various syntactic changes that need to be made inside all 4 XML files.

You will notice that the DTD declaration inside **"`amazon-extended.xml`"** needs to be removed. Why?
Are there any other changes needed to take place inside the 3 XML files?
Tip: If you associate the 3 XML files with the **"`amazon.dtd`"**, a slight modification will be necessary in the initial XML. What modification would that be?
When all necessary changes take place, you will have a centrally stored source data repository (**"`amazon-extended.xml`"**) and various views of it (the 3 XML files), each with its own presentation style.

2. Create a simple frame-based website for presenting the content you developed previously. The website will have a navigation frame on the left and another frame for displaying the content on the right. The navigation frame contains a list of links with the second frame as the target. The second frame will show the linked document.

You will need to develop an HTML file that describes the two frames and another HTML file for the left frame. The frame on the right will always be one of the 3 XML files, so it is not necessary to develop an additional HTML page for it.
Tip: Check the following link for examples on HTML frames (including navigation frames): http://www.w3schools.com/html/html_frames.asp

3. What series of actions is now necessary, in order to easily update the catalog of products? (e.g. add new products to the list)