****

**APT 1040 Introduction to Web Design and applications**

**Lab 6: Writing a Simple XML File**

Let's start out by writing up a simple version of the kind of XML data you could use for a slide presentation. In this exercise, you'll use your text editor to create the data in order to become comfortable with the basic format of an XML file. You'll be using this file and extending it in later exercises.

**Creating the File**

Using a standard text editor, create a file called slideSample.xml.

**Writing the Declaration**

Next, write the declaration, which identifies the file as an XML document. The declaration starts with the characters "<?", which is the standard XML identifier for a *processing instruction*. (You'll see other processing instructions later on in this tutorial.)

<?xml version='1.0' encoding='utf-8'?>

This line identifies the document as an XML document that conforms to version 1.0 of the XML specification, and says that it uses the 8-bit Unicode character-encoding scheme. (For information on encoding schemes, see [Java Encoding Schemes](http://www.inf.fu-berlin.de/lehre/SS03/19560-P/Docs/JWSDP/tutorial/doc/Encodings.html#64176).)

Since the document has not been specified as "standalone", the parser assumes that it may contain references to other documents. To see how to specify a document as "standalone", see [The XML Prolog](http://www.inf.fu-berlin.de/lehre/SS03/19560-P/Docs/JWSDP/tutorial/doc/IntroXML2.html#63920).

**Adding a Comment**

Comments are ignored by XML parsers. You never see them in fact, unless you activate special settings in the parser. You'll see how to do that later on in the tutorial, when we discuss [Handling Lexical Events](http://www.inf.fu-berlin.de/lehre/SS03/19560-P/Docs/JWSDP/tutorial/doc/JAXPSAX16.html#65510). For now, add the text highlighted below to put a comment into the file.

<?xml version='1.0' encoding='utf-8'?>

**<!-- A SAMPLE set of slides -->**

**Defining the Root Element**

After the declaration, every XML file defines exactly one element, known as the root element. Any other elements in the file are contained within that element. Enter the text highlighted below to define the root element for this file, slideshow:

<?xml version='1.0' encoding='utf-8'?>

<!-- A SAMPLE set of slides -->

**<slideshow>**

**</slideshow>**

Note: XML element names are case-sensitive. The end-tag must exactly match the start-tag.

**Adding Attributes to an Element**

A slide presentation has a number of associated data items, none of which require any structure. So it is natural to define them as attributes of the slideshow element. Add the text highlighted below to set up some attributes:

...

  <slideshow

**title="Sample Slide Show"**

**date="Date of publication"**

**author="Yours Truly"**

    >

  </slideshow>

When you create a name for a tag or an attribute, you can use hyphens ("-"), underscores ("\_"), colons (":"), and periods (".") in addition to characters and numbers. Unlike HTML, values for XML attributes are always in quotation marks, and multiple attributes are never separated by commas.

Note: Colons should be used with care or avoided altogether, because they are used when defining the namespace for an XML document.

**Adding Nested Elements**

XML allows for hierarchically structured data, which means that an element can contain other elements. Add the text highlighted below to define a slide element and a title element contained within it:

<slideshow

  ...

  >

**<!-- TITLE SLIDE -->**

**<slide type="all">**

**<title>Wake up to WonderWidgets!</title>**

**</slide>**

</slideshow>

Here you have also added a type attribute to the slide. The idea of this attribute is that slides could be earmarked for a mostly technical or mostly executive audience with type="tech" or type="exec", or identified as suitable for both with type="all".

More importantly, though, this example illustrates the difference between things that are more usefully defined as elements (the title element) and things that are more suitable as attributes (the type attribute). The visibility heuristic is primarily at work here. The title is something the audience will see. So it is an element. The type, on the other hand, is something that never gets presented, so it is an attribute. Another way to think about that distinction is that an element is a container, like a bottle. The type is a characteristic of the *container* (is it tall or short, wide or narrow). The title is a characteristic of the *contents* (water, milk, or tea). These are not hard and fast rules, of course, but they can help when you design your own XML structures.

**Adding HTML-Style Text**

Since XML lets you define any tags you want, it makes sense to define a set of tags that look like HTML. The XHTML standard does exactly that, in fact. You'll see more about that towards the end of the SAX tutorial. For now, type the text highlighted below to define a slide with a couple of list item entries that use an HTML-style <em> tag for emphasis (usually rendered as italicized text):

  ...

  <!-- TITLE SLIDE -->

  <slide type="all">

    <title>Wake up to WonderWidgets!</title>

  </slide>

**<!-- OVERVIEW -->**

**<slide type="all">**

**<title>Overview</title>**

**<item>Why <em>WonderWidgets</em> are great</item>**

**<item>Who <em>buys</em> WonderWidgets</item>**

**</slide>**

</slideshow>

We'll see later that defining a *title* element conflicts with the XHTML element that uses the same name. We'll discuss the mechanism that produces the conflict (the DTD) and several possible solutions when we cover [Parsing the Parameterized DTD](http://www.inf.fu-berlin.de/lehre/SS03/19560-P/Docs/JWSDP/tutorial/doc/JAXPSAX15.html#65449).

**Adding an Empty Element**

One major difference between HTML and XML, though, is that all XML must be *well-formed* -- which means that every tag must have an ending tag or be an empty tag. You're getting pretty comfortable with ending tags, by now. Add the text highlighted below to define an empty list item element with no contents:

  ...

  <!-- OVERVIEW -->

  <slide type="all">

    <title>Overview</title>

    <item>Why <em>WonderWidgets</em> are great</item>

**<item/>**

    <item>Who <em>buys</em> WonderWidgets</item>

  </slide>

</slideshow>

Note that any element can be empty element. All it takes is ending the tag with "/>" instead of ">". You could do the same thing by entering <item></item>, which is equivalent.

Note: Another factor that makes an XML file *well-formed* is proper nesting. So <b><i>some\_text</i></b> is well-formed, because the <i>...</i> sequence is completely nested within the <b>..</b> tag. This sequence, on the other hand, is not well-formed: <b><i>some\_text</b></i>. 

**The Finished Product**

Here is the completed version of the XML file:

<?xml version='1.0' encoding='utf-8'?>

<!-- A SAMPLE set of slides -->

<slideshow

  title="Sample Slide Show"

  date="Date of publication"

  author="Yours Truly"

  >

  <!-- TITLE SLIDE -->

  <slide type="all">

    <title>Wake up to WonderWidgets!</title>

  </slide>

  <!-- OVERVIEW -->

  <slide type="all">

    <title>Overview</title>

    <item>Why <em>WonderWidgets</em> are great</item>

    <item/>

    <item>Who <em>buys</em> WonderWidgets</item>

  </slide

</slideshow>

Try opening it in a web browser