XML - Tags, Elements, Content, and Attributes*

R.G. (Dick) Baldwin

Abstract

This tutorial lesson is part of a series dedicated to programming with Adobe Flex. Flex is a programming language based on XML. Therefore, in order to program with Flex, you must first understand XML. The previous lesson, this lesson, and lessons that follow provide a brief introduction to XML.

1 Table of Contents

2 Preface

2.1 General

This tutorial lesson is part of a series dedicated to programming with Adobe Flex.

Flex is a programming language based on XML. Therefore, in order to program with Flex, you must first understand XML. The previous lesson, this lesson, and lessons that follow provide a brief introduction to XML.

2.2 Viewing tip

I recommend that you open another copy of this document in a separate browser window and use the following links to easily find and view the figures and listings while you are reading about them.

*Version 1.1: Jun 8, 2010 6:31 pm +0000

†http://creativecommons.org/licenses/by/3.0/

2.2.1 Figures

- Figure 1 (p. 3) . Tag example.

2.2.2 Listings

- Listing 1 (p. 2) . Simple Flex MXML code.
- Listing 2 (p. 3) . An element containing tags, content, and an attribute.
- Listing 3 (p. 4) . A section of raw XHTML code.
- Listing 4 (p. 5) . Nested elements.

2.3 Supplemental material

I recommend that you also study the other lessons in my extensive collection of online programming tutorials. You will find a consolidated index at www.DickBaldwin.com [1] .

3 Tags, elements, content, and attributes

Listing 1 shows the code from a very simple Flex MXML file. *(Note that the code shown in Listing 1 is from Flex version 3.x. Code from version 4.x would be different in several respects.)*

Listing 1: Simple Flex MXML code.

```
    <?xml version="1.0" encoding="utf-8"?>
<!--DragAndDrop04-->

<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:cc="CustomClasses.*">
  <cc:Driver/>

</mx:Application>
```

I'm not going to explain the MXML code in this lesson. I am simply providing the code as an example of an XML document so you can see what a real XML document looks like.

What is a tag?

I will refer to items enclosed in angle brackets, such as those shown in Figure 1, as tags.

[1] http://www.dickbaldwin.com/toc.htm

Tag example.

```
<book>

...

</book>
```

Figure 1: Tag example.

The first tag shown in Figure 1 is often referred to as a *start tag* . The second tag is often referred to as an *end tag* .

Note that the start tag and the end tag differ only in that the end tag contains a slash character. However, the start tag can also contain optional namespace indicators and attributes as discussed below.

What are elements, content, and attributes?

Listing 2 contains a *start tag* and an *end tag* with an *attribute* and some *content* .

Listing 2: An element containing tags, content, and an attribute.

```
  <chap number="1">
Text for Chapter 1
</chap>
```

An element

The entire set of characters beginning with the start tag and ending with the end tag constitutes an element [2] .

An element usually consists of a start tag and an end tag with the content sandwiched in between the two tags, but there are exceptions to that rule. You will learn about those exceptions, including empty elements later.

The tags

You have probably already recognized the tags in Listing 2 as the two sets of characters beginning with a left angle bracket and ending with a right angle bracket.

The start tag may contain optional attributes. In Listing 2, a single attribute provides the *number* value for the chapter.

The start tag may also contain *namespace* information. There is no namespace information in Listing 2. You will learn about namespaces in a future lesson.

The content

The set of characters in between the tags constitutes the *content* .

An attribute

The set of characters following the word chap in the start tag constitutes an *attribute* .

[2] http://www.w3.org/TR/REC-xml/#sec-starttags

The term attribute is commonly used in computer science and usually has about the same meaning, regardless of whether the discussion revolves around XML, Java programming, or database management. An attribute often serves to partially describe the thing to which it refers.

Things have attributes

A chapter in a book is a thing and a chapter has attributes such as its number. A person is also a thing. Therefore, a person also has attributes. Each attribute has a value. Here is a list of some of the attributes *(along with their values)* that might be used to describe a person:

- name="Joe"
- height="84"
- weight="176"
- complexion="pale"
- sex="male"
- training="Java programmer"
- degree="Masters"

Obviously, there are many more attributes that could be used to describe a person.

The importance of an attribute depends on the context

Whether or not a particular attribute for a person is important depends on the context in which the person is being considered. For example, if the person is being considered in the context of a candidate for a basketball team, the height, weight, and sex attributes will probably be important.

On the other hand, if the person is being considered in the context of a candidate for employment as a computer programmer, the height, weight, and sex attributes should not be important at all, but the training and degree attributes might be very important.

Why does XML use attributes?

The description of XML that I provided in an earlier lesson is repeated here for convenience:

> *"XML gives us a way to create and maintain structured documents in plain text that can be rendered in a variety of different ways."*

Attributes often provide information that is needed for the rendering process, but attributes have many other uses as well.

Rendering

I suggested earlier that the most common modern use of the word rendering means to present something for human consumption. I gave an example of a newspaper that can either be rendered on newsprint paper or can be rendered on a computer screen.

A rendering engine

If the newspaper *(structured document)* is created and maintained as an XML document, then some sort of computer program *(often referred to as a rendering engine)* will probably be used to render it into the desired presentation format.

This document

For example, the original version of this document was created as a special flavor of an XML document known as an XHTML document. Listing 3 shows a short sample of the raw XML code in the original document.

Listing 3: A section of raw XHTML code.

```
    <p> <strong style="color:#ff0000" ">A rendering
engine  "</strong> </p>
<p>If the newspaper (structured document) is
created and maintained as an XML document, then
some sort of computer program (often referred to
```

```
as a rendering engine) will probably be used to
render it into the desired presentation format. </p>
 <p>For example, this document is a special flavor
of an XML document known as an XHTML document.
Listing 3 shows a short sample of the raw XML
code that was delivered to your computer from my
website.  </p>
```

Created using a WYSIWYG XHTML editor

I originally created the document as an XHTML document using a WYSIWYG XHTML editor that behaves much like a word processor. *(If you don't know what WYSIWYG means, Google it.)* Of course, the document has since undergone quite a lot of editing so the final XHTML version probably doesn't match the XHTML code in Listing 3.

Transform to CNXML

Later on, I used a Java program of my own design to transform the final XHTML document into another flavor of XML known as CNXML for publishing on the Connexions [3] website.

That illustrates another characteristic of XML. Because the formats of certain flavors of XML documents are well defined, it is often practical to transform them from one flavor to another flavor.

That makes it possible for me to create the document using a program that is very similar to a word processor and then transform the output of that program into a fairly cryptic format that satisfies the publishing requirements of the website.

Your browser is rendering the document

When you accessed the document from the Connexions [4] website, it was transformed back into an XHTML document and sent to your computer.

As you can see in Listing 3, viewing raw XHTML isn't very enjoyable. Fortunately, your browser is acting as a rendering engine to render the raw XHTML text into a much more pleasing presentation format.

Back to the book example

A book that is created and maintained as an XML document could be rendered in a variety of different ways. Whichever way it is rendered, however, it would probably be useful to separate and number the chapters. Therefore, the value of the *number* attribute could be used by the rendering engine to present the chapter number for a specific rendering.

In some renderings, the number might appear on an otherwise blank page that begins a new chapter. In a different rendering, the chapter number might appear in the upper right or left-hand corner of each page.

Tell me again, what is an element?

As I explained earlier, an *element* usually consists of *start tag (with optional attributes and namespace information)* , an *end tag* , and the *content* in between as shown earlier in Listing 2.

Elements can be nested

Elements can be nested inside other elements as shown in Listing 4.

Listing 4: Nested elements.

```
   <book>
  <chapter number="1">
    Content for Chapter 1
  </chapter>

  <chapter number="2">
    Content for Chapter 2
```

[3] http://cnx.org/content/col11207/latest/
[4] http://cnx.org/content/col11207/latest/

```
  </chapter>
</book>
```

In Listing 4, two chapter elements are nested inside a book element.

Why does XML use elements?

It is probably fair to say that the element constitutes the fundamental unit of information in an XML document. For example, the element defines the type of information, such as chapter in our book example.

Sandwiched in between the start tag and the end tag of an element, we find the raw information *(content)* that the XML document is designed to convey. For a text document, you are likely to find a lot of content between the tags. For example, in Listing 3, there are several lines of text between the paragraph tags identified by the p and the /p enclosed in angle brackets.

Once again, what is content?

Of the four terms mentioned earlier, *(tags, elements, content, and attributes)* , content is the easy one. Content is sandwiched in between the start tag and the end tag of an element. Usually the content of the elements contains the information that the XML document is designed to convey. In other words, this is where we put the information for which the document was created. The tags and attributes are there to create the structure.

For example, if the XML document is being used for the creation and maintenance of material for a newspaper, the content constitutes the news. If the XML document is being used for the creation and maintenance of a Java programming textbook, the content contains the information about Java programming that we want to convey to the student.

Why do we need structure?

One of the primary objectives of XML is to separate content from presentation. If we insert the raw material as content into a structure defined by the tags, elements, and attributes, then that raw material can be presented in a variety of ways.

4 Resources

I will publish a list containing links to Flex resources as a separate document. Search for Flex Resources in the Connexions search box.

5 Miscellaneous

This section contains a variety of miscellaneous materials.

NOTE: **Housekeeping material**

- Module name: XML - Tags, Elements, Content, and Attributes
- Files:
  - Flex0082\Flex0082.htm
  - Flex0082\Connexions\FlexXhtml0082.htm

-end-