XML - Well-Formed and Valid Documents*

Richard Baldwin

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

Abstract

This tutorial lesson is part of a continuing 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. In this lesson, you will learn about well-formed documents, valid documents, and the DTD.

1 Table of Contents

- Preface (p. 1)
  - General (p. 1)
  - Viewing tip (p. 2)
    - Figures (p. 2)
    - Listings (p. 2)
  - Supplemental material (p. 2)
- Well-formed and valid documents (p. 2)
  - Valid documents and the DTD (p. 2)
  - Well-formed documents (p. 6)
  - Validity and well-formed requirements recap (p. 8)
- Resources (p. 8)
- Miscellaneous (p. 8)

2 Preface

2.1 General

This tutorial lesson is part of a continuing 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.

*Version 1.1: Jun 10, 2010 3:53 pm -0500

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

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.

2.2.1 Figures

- Figure 1 (p. 3) . What is a DTD?
- Figure 2 (p. 7) . Why do we need well-formed XML documents?

2.2.2 Listings

- Listing 1 (p. 5) . Raw XHTML code.
- Listing 2 (p. 5) . A small portion of the XHTML DTD.
- Listing 3 (p. 7) . Required syntax for an empty element.

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 Well-formed and valid documents

In previous lessons, I have discussed tags, elements, content, and attributes in detail. The time has come to take up the following topics:

- Well-formed documents
- Valid documents
- The DTD

3.1 Valid documents and the DTD

What is a DTD?

Figure 1 contains a quotation from the XML FAQ [2] that describes a DTD.

[1] http://www.dickbaldwin.com/toc.htm
[2] http://xml.silmaril.ie/

What is a DTD?

```
    "A DTD is usually a file (or several files to be used
together) which contains a formal definition of a particular
type of document. This sets out what names can be used for
elements, where they may occur, and how they all fit together.
For example, if you want a document type to describe <LIST>s
which contain <ITEM>s, part of your DTD would contain
something like

<!ELEMENT item (#pcdata)>
<!ELEMENT list (item)+>

This defines items containing text, and lists containing
items.

It's a formal language which lets processors automatically
parse a document and identify where every element comes and
how they relate to each other, so that stylesheets, navigators,
browsers, search engines, databases, printing routines, and
other applications can be used."
```

Figure 1: What is a DTD?

DTDs are complicated

I included the above quotation to emphasize one very important point – DTDs are complicated. The creation of a DTD of any significance is a very complex task.

The good news!

The good news is that many of you will never need to worry about having to create a DTD for two reasons:

1. In the most fundamental sense, XML does not require the use of a DTD.
2. Even when it is advisable to use a DTD with XML, someone else may already have created the DTD on your behalf.

A validating XHTML editor

For example, I wrote the original version of this HTML document using a validating XHTML editor named Amaya [3] . Even though the editor uses a DTD to confirm that my document is a valid XHTML document *(and warns me if it isn't)* , it wasn't necessary for me to write the DTD. The people who wrote the editor also wrote the DTD.

Three Parts

[3] http://www.w3.org/Amaya/

It is reasonable to think of an XML document as consisting of three parts, some of which are optional. I'm gong to refer to the parts as files just so I will have something to call them *(but they don't have to be separate physical files)* .

One file contains the information content of the document *(words, pictures, etc.)* . This is the part containing tags, elements, content, and attributes that the author wants to expose to the client. I have discussed this part in previous lessons.

A second file is the DTD, which meets the definition given above.

A third file is a stylesheet that establishes how the content that conforms to the DTD is to be rendered on the output device. This is how the author wants the material to be presented to the client.

Rendering

For example a tag with an attribute of "red" might cause something to be presented bright red according to one stylesheet and dull red according to another stylesheet. *(It might even be presented as some shade of green according to still another stylesheet.)*

With XML, the DTD is optional but the stylesheet (or some processing mechanism that substitutes for a stylesheet) is generally required. Something has to be able to render the content in the manner that the author intended it to be rendered. Otherwise, the client will be forced to view the document as raw XML text, which usually isn't very enjoyable.

A DTD can be very complex

Once again, according to the **XML FAQ** :

> NOTE: "... the design and construction of a DTD can be a complex and non-trivial task, so XML has been designed so it can be used either with or without a DTD. DTDless operation means you can invent markup without having to define it formally. To make this work, a DTDless file in effect 'defines' its own markup, informally, by the existence and location of elements where you create them. But when an XML application such as a browser encounters a DTDless file, it needs to be able to understand the document structure as it reads it, because it has no DTD to tell it what to expect, so some changes have been made to the rules."

Without the technical jargon please

In other words, it is entirely possible to create an XML document without the requirement for a DTD.

What is a valid document?

In the normal sense of the word, if something is *invalid* , that usually means that it is not any good. However, that is not the case for XML. An invalid XML document can be a perfectly good and useful document.

A valid XML document is one that conforms to an existing DTD in every respect.

In other words, unless the DTD allows a tag with the name "color", an XML document being validated against that DTD containing a tag with that name is not valid.

However, because XML does not require a DTD, an XML processor cannot require validation of the document. Many very useful XML documents are not valid, simply because they were not constructed according to an existing DTD.

An XHTML document

The document that you are now reading was originally created as a valid XML document before being transformed to CNXML and uploaded to the Connexions website. It was created as a special flavor of XML known as XHTML. As I mentioned earlier, the document was created using W3C's WYSIWYG Editor/Browser named Amaya [4] .

What you are probably reading now is a rendered version of the document after having gone through a couple of transformations. However, if you were to have looked at the raw XHTML code at the beginning of the document before it was transformed to CNXML, you would have seen something like the XML code shown in Listing 1.

[4] http://www.w3.org/Amaya/

Listing 1: Raw XHTML code.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type"
content="text/html; charset=iso-8859-1" />
<title>Flex Programming by Richard G. Baldwin</title>
<meta name="generator"
content="Amaya, see http://www.w3.org/Amaya/" />
</head>
```

(Note that some extra line breaks were inserted in Listing 1 to force it to fit into this narrow publication format.)

The DTD

Note in particular the code that begins with "http: in Listing 1. This code specifies the DTD that is used to validate the XML code. If I had inadvertently entered some XML code that caused the document to become invalid, a red warning would have appeared in the bottom right corner of the Amaya editor.

A download site

If you examine the DTD information in Listing 1 carefully, you will see that it actually specifies a location on the Internet from which you can download the DTD file. You can download it and open it in a text editor such as Windows Notepad to see a sample of a really complicated DTD.

Listing 2 shows a small portion of the XHTML DTD downloaded from the address shown in Listing 1.

Listing 2: A small portion of the XHTML DTD.

```
<!--
Extensible HTML version 1.0 Transitional DTD

This is the same as HTML 4 Transitional except for
changes due to the differences between XML and SGML.

Namespace = http://www.w3.org/1999/xhtml

For further information, see: http://www.w3.org/TR/
xhtml1

Copyright (c) 1998-2002 W3C (MIT, INRIA, Keio),
All Rights Reserved.

This DTD module is identified by the PUBLIC and
SYSTEM identifiers:

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd"
```

```
   $Revision: 1.2 $
   $Date: 2009-12-14$

-->

<!-------- Character mnemonic entities ==============-->

<!ENTITY % HTMLlat1 PUBLIC
   "-//W3C//ENTITIES Latin 1 for XHTML//EN"
   "xhtml-lat1.ent">
%HTMLlat1;

<!ENTITY % HTMLsymbol PUBLIC
   "-//W3C//ENTITIES Symbols for XHTML//EN"
   "xhtml-symbol.ent">
%HTMLsymbol;

<!ENTITY % HTMLspecial PUBLIC
   "-//W3C//ENTITIES Special for XHTML//EN"
   "xhtml-special.ent">
%HTMLspecial;
```

(Once again, I inserted some line breaks into the text in Listing 2 to force it to fit into this publication format.)

3.2 Well-formed documents

XML derives from an earlier more complicated markup language known as SGML. Being well-formed is not a property of SGML. The concept of being well-formed was introduced as a requirement of XML, apparently to deal with the situation where a DTD is not available.

Why do we need well-formed XML documents?

Once again, according to the XML FAQ [5] :

[5] http://xml.silmaril.ie/

Why do we need well-formed XML documents?

```
    "For example, HTML's <img> element is defined as 'EMPTY':
it doesn't have an end-tag. Without a DTD, an XML application
would have no way to know whether or not to expect an end-tag
for an element, so the concept of 'well-formed' has been
introduced.

This makes the start and end of every element, and the
occurrence of EMPTY elements completely unambiguous."
```

Figure 2: Why do we need well-formed XML documents?

All XML documents must be well-formed

XML documents need not be valid, but

ALL XML DOCUMENTS MUST BE WELL-FORMED.

To be well-formed...

A well-formed XML document must meet several different criteria.

To begin with, in a well-formed XML document, all elements that can contain character data must have both start and end tags.

What is character data?

For purposes of this explanation, let's just say that the content that we discussed earlier comprises character data.

Attribute values must be in quotes

All attribute values must be in quotes *(apostrophes or double quotes)* . You can surround the value with apostrophes *(single quotes)* if the attribute value contains a double quote. An attribute value that is surrounded by double quotes can contain apostrophes.

Dealing with empty elements

EMPTY elements *(those that contain no character data)* must be written in one of the two ways shown in Listing 3, and for several reasons, the first way is usually considered preferable.

Listing 3: Required syntax for an empty element.

```
    <mx:Button label="My button."/>
<mx:Button label="My button."></mx:Button>
```

Don't forget that even an EMPTY element can contain one or more attributes along with namespace information inside the start tag. *(In the case of Listing 3,* ***mx:*** *is namespace information and the* ***label*** *information is an attribute.)*

Markup characters and entities

There are also rules regarding the inclusion of markup characters.

NOTE: **No markup characters are allowed**

```
    For a document to be well-formed, it must not have markup
characters such as angle brackets or ampersands in the text
data. If such characters are needed, you can represent them
using &lt; and &amp; instead.

These special combinations of characters that represent other
characters, such as &lt; that represents the left angle
bracket are called entities.
```

Nesting

Elements must nest properly. If one element contains another element, the entire second element must be defined inside the start and end tags of the first element. In the next lesson, you will learn that every element in an XML document, other than the root element, is nested inside another element.

3.3 Validity and well-formed requirements recap

Valid XML files are those that have a DTD and that conform to the DTD.

All XML files must be well-formed, but there is no requirement for them to be valid.

A DTD is not required in which case validity is impossible to establish. However, if XML documents do have a DTD, they must conform to it, which makes them valid.

Why use a DTD if it is not required?

There are many reasons to use a DTD, in spite of the fact that XML doesn't require one. One reason is that the use of a DTD makes it possible to enforce format specifications. For example, in a document that represents a book, the DTD could require that paragraph elements can occur only inside of page elements. It could also require that page elements can occur only inside chapter elements. It could require that there be a preface element and that it must occur before any chapter elements.

Enforcing format specifications

For example, by creating this document using Amaya and the DTD for XHTML, I was required to produce a document that conformed to the DTD for XHTML documents. Otherwise, I would have gotten warnings from the editor and would have been required to acknowledge that the document didn't conform to the DTD in order to save it.

On one hand, that sounds like a lot of hassle. On the other hand, by creating a document that conforms to the DTD for XHTML, I can be sure that it will render properly in any browser that is guaranteed to properly render XHTML documents.

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 - Well-Formed and Valid Documents
- Files:
  - Flex0084\Flex0084.htm
  - Flex0084\Connexions\FlexXhtml0084.htm

NOTE: **PDF disclaimer:** Although the Connexions site makes it possible for you to download a PDF file for this module at no charge, and also makes it possible for you to purchase a pre-printed version of the PDF file, you should be aware that some of the HTML elements in this module may not translate well into PDF.

-end-