



XML Primer

Author: Scott Cadillac

Date: Monday, June 17, 2002

Email: scott@xml-extra.net

Basic XML is Well-formed XML

Extensible Markup Language (XML) is often characterized as "self describing" and has no pre-defined elements (also known as "tags") - but that doesn't mean you can just invent your own rules on how to use XML.

XML is popularly known as a "data markup language". And when XML is coupled with one of its dialects or spin-off technologies, e.g., XSLT, XML can be capable of providing a wide range of meaningful information and functionality with data and/or presentation languages like HTML. But in the context of having a starting point in learning XML, we need to understand XML basics. And all XML documents of every description have to adhere to some simple primary rules in order to qualify the XML as valid and *valid XML is known as "well-formed XML"*.

Basic Well-formed XML Rules

- Must have only a single root element
- All elements require a closing tag or ending slash
- Element names are case-sensitive
- Properly nest all elements
- Double-quote all attribute values
- Use reserved character entities or use `<![CDATA[]]>` sections

Following is a simple example of well-formed XML. From this example we'll identify the basic parts mentioned in the well-formed rules, which make valid XML.

Good, Well-formed XML

```
<?xml version="1.0" ?>
<myAddressBook>
  <contact ID="345">
    <name>Elmore Fudd</name>
    <title>Farmer</title>
    <email>leatherfreak@hotmail.com</email>
    <company>Bugs & Warber Bros. Holdings</company>
  </contact>
  <contact ID="678">
    <name>Tweety</name>
    <title>Bird</title>
    <email>piercedude@aol.com</email>
    <company/>
  </contact>
</myAddressBook>
```

Note: The first line of the above example contains a processing instruction, *i.e.*, `<?xml version="1.0" ?>` and should not be confused with normal XML elements. XML Processing instructions, declarations or comments start with '`<?'`', '`<!--'`' or '`<!--'`'.

An **"element"** is an identifier or marker in your XML text which is intended to represent the meaning and placement of a piece of data or a collection of information. Elements are easy to spot because they are surrounded by the less-than '`<`' and greater-than '`>`' characters - also known as angle brackets.

Example `<company>`.

- Element names can contain letters, numbers, underscores and some other characters - but never use spaces.
- Element names should begin with a letter - but not begin with "xml" (in any case combination).
- Element names can be descriptive - but avoid really long names.
- Elements are sometimes referred to as "tags" and are also known as "nodes".
- Element names with international characters should not be used unless the the proper processing instructions are included (not covered in this article).
- Element names with the colon ':' character are intended for use with XML Namespaces (not covered in this article).

■ Must have only a single root element

- The "root element" is the base or beginning of your XML. An XML document and its information is often characterized as "hierarchical" in nature, which means the very top of the hierarchy is also the "root" - therefore there can only be one "root".
- In the well-formed XML example above, `<myAddressBook>` is the root element.

An XML document is any collection of valid XML contained within a single root element - whether generated dynamically in an application or a programmed function or as a stand alone text file. There are no limits to the size of an XML document and it can be as small as an empty root element, e.g., `<myAddressBook/>`.

■ All elements require a closing tag or ending slash

- Elements are intended to represent information. Typically this information is "contained" within a starting element and an ending element, e.g., `<title>Farmer</title>` ("title" is the element name and "Farmer" is the information).
- The "beginning element" is the information identifier shown simply as `<title>` and can also contain attributes (described later). The "ending element" is a repeat of the beginning element, but the element name is preceded with a slash '/' character, i.e., `</title>`. An ending element will not contain attributes.
- Elements that contain no information between the beginning and ending elements are considered "empty". Empty elements can be represented two ways:
 - `<title></title>` is a valid empty element.
 - `<title/>` is also a valid empty element. Note the placement of the ending slash '/' and the absence of the ending element - this is a shortcut method for representing an empty element.
- **Note:** An empty element can still contain information by using attributes, e.g., `<title name="Farmer"/>`.

■ Element names are case-sensitive

- The unique and descriptive names given to your elements are case-sensitive. For example, when navigating an XML document, the `<title/>` element is referred to as "title" - not "Title".
- Likewise, the letter-case use of the name of the beginning and ending elements must match. For example, `<title>Farmer</title>` observes case-sensitive element names correctly, but `<Title>Farmer</title>` does not.

■ Properly nest all elements

- Nested elements are elements that do not overlap. Although element can exist within one another - at the end of each element's information, it must be marked with it's ending element before starting a new element.
 - These elements are correctly nested:

```
<rootNode>
  <name>Elmore Fudd</name>
  <title>Farmer</title>
</rootNode>
```

- Below, the <name> and <title> elements are incorrectly nested:

```
<rootNode>
  <name>Elmore Fudd<title>
  </name>Farmer</title>
</rootNode>
```

The use of "attributes" with an element is an optional method of data representation. Although attributes typically only provide a single piece of data, an element with several attributes combined with the element's value can represent complex information.

Example of an element with attributes: <contact ID="678" name="Tweety" title="Bird"/>

Note: Attributes names are separated from the element names by space or tab characters.

- Double-quote all attribute values**

- HTML is often forgiving when it comes to quoting attribute values - but the XML specification simply does not allow it. Example, <contact ID="678"/> has the ' ID ' attribute and the value ' 678 ' properly double-quoted.
- Although both single ' ' and double-quotes ' " ' are allowed - it is widely practiced that double-quotes are the standard.
- Witango Note:** The XML DOM support in Witango 4.0 and higher, prefers double-quotes.

Note: Attributes are important because these can become the more efficient way of navigating to a particular element and it's information, within your XML document.

- Use reserved character entities or use <![CDATA[]]> sections**

- "Entities" are a method of representing a character or a string with a special character code combination, which always starts with an ampersand ' & ' and ends with a semi-colon ' ; '.
- In all XML languages, the following characters are reserved, and should be "escaped" by being represented via their special entity:

Character	Name	Entity alternative
<	less-than	<
>	greater-than	>
&	ampersand	&
'	single-quote	'
"	double-quote	"

- An alternative to entities is to enclose the information within CDATA sections. A CDATA section is a special method for marking text that is to be ignored by whatever XML validation process the document may encounter.
 - The beginning of a CDATA section starts with ' <![CDATA[' and ends with ']]> '.
 - CDATA sections are only used between a beginning and ending element and not with attributes.
- The following examples are for demonstrating the use of reserved characters in XML:
 - <comment>50 is < 80 & > 20</comment> will cause a validation error.
 - <comment>50 is < 80 & > 20</comment> is valid XML.
 - <comment><![CDATA[50 is < 80 & > 20]]></comment> is valid XML.
 - <comment><![CDATA[50 is < 80 & > 20]]></comment> is valid XML.

B a d X M L

Near the top of this page is a simple example of well-formed XML. Directly below is another simple example of XML which is not well-formed. By comparing the two, we can examine what makes them different, to help better understand what is well-formed.

Bad, invalid XML

```

1 <?xml version="1.0">
2 <myaddressbook>
3     <contact ID=345>
4         <name>Elmore Fudd<title></name>Farmer</title>
5         <email>leatherfreak@hotmail.com</email>
6         <company>Bugs & Warber Bros. Holdings
7     </contact>
8     <contact ID=678>
9         <name>Tweety</Name>
10        <title>Bird</title>
11        <Email>piercedude@aol.com</email>
12        <company>
13    </contact>
14 </myAddressBook>

```

Note: The **coloured** line numbers in the left margin have been added to the example above to help identify what makes the XML invalid.

What's wrong with the bad, invalid XML above

- Line **2**, the name of the beginning root element node has different case letters than the ending root element node (line **14**).
 - Line **3** and line **8**, the ID attribute values are not double-quoted.
 - Line **4**, the `<name>` and `<title>` elements are not properly nested.
 - Line **6**, the `<company>` element is missing an ending `</company>` element.
 - Line **6** again, the ampersand character `&` is not escaped or the string is not surrounded with `<![CDATA[]]>`.
 - Line **9**, the case letters for the `<name>` element do not match.
 - Line **11**, the case letters for the `<email>` element do not match.
 - Line **12**, the `<company>` element has no inner value. Empty elements like this should either be closed with a matching ending element or can be short-cut with an ending slash, *e.g.*, `<company/>`.
 - And lastly, line **1**. Although this is a processing instruction (not an element) and is optional in some cases - it is missing a closing question mark `'?`, *i.e.*, `<?xml version="1.0" ?>`
- [Manuel élémentaire XML \(French Translation\)](#)
 - [Also see, XPointer with Witango...](#)

[Copyright, disclaimers, feedback and about this site.](#)

[<root>](#) | [<witango>](#) | [<msie>](#) | [<xml>](#) | [<web-services>](#) | [<web-tools>](#) | [<salsa>](#) | [<surf>](#) | [</root>](#)

Last site update: Wednesday, October 30 2002 02:19 PM PST

Web-master: [Scott Cadillac](#)