

XML basics

This chapter introduces the basic requirements of XML code. It illustrates by example how to create a XML document with correctly formed syntax and demonstrates how a complete XML document can be formatted for display in a web browser with a stylesheet.

Covers

What is XML? | 20

Tag format | 21

The XML declaration | 22

Elements | 23

Element attributes | 24

Empty elements | 25

Comments | 26

Character entities | 27

CDATA blocks | 28

XML example document | 29

Example document output | 30

Chapter Two

What is XML?

Like most good ideas, XML is basically a very simple idea that can be put to good use in many situations. It is not intended to be a direct replacement for the general-purpose markup that is provided by HTML describing how content should be displayed. Instead XML offers a means to define and construct other custom markup languages with tags you name yourself and rules which you define for how those tags can be used. So because XML is a language that describes other languages it is termed as being a “meta-language”.

The two words that should be most strongly associated with any definition of XML are “structure” and “data”.

Custom markup languages constructed in XML will arrange data in a structured manner so it can be accessed and manipulated in a variety of ways. For instance, you could use XML to create a custom markup language called “EruptML” to describe the data on all active volcanoes.

Technically the custom markup languages that are created in XML are known as “XML applications” but, to avoid confusion with normal software applications, that term is not used in this book.

The mechanics of making up your own markup tags are covered in this chapter of the book together with an explanation of the syntax rules that must be obeyed to ensure that your XML document has well-formed code.

Following after this basic grounding are chapters explaining how to define rules for your tags with a Document Type Definition (DTD) or with the more comprehensive XMLSchema language.

Finally, after learning how to create a well-formed XML document with custom tags that are valid to your defined rules, we can look at how the structured data in the document can actually be used.

Using data in a XML document often employs other XML technologies which are explored in the later chapters in the book.

But first, discover the basics of XML in this chapter...

Tag format

XML element tags look much like those you see in HTML but, unlike HTML, there are no pre-defined elements. Greater care must be taken with XML elements to ensure that they adhere to strict syntax rules to make the XML document well-formed.

Case

XML is a case-sensitive language, so for instance the tags `<address>`, `<Address>` and `<ADDRESS>` are considered to be three separate unrelated elements in XML. It is recommended that you use only lowercase for element names to avoid confusion.

Naming conventions

Element names in XML may only start with either a letter or underscore character. The rest of the name may consist of any mixture of letter, number, underscore, dot or hyphen characters. Spaces are not allowed in element names nor can the name begin with the string “xml” which is reserved for the XML specification. `<mytag>`, `<_mytag>` and `<my-tag1.extra>` are all valid names. `<1mytag>`, `<my tag>` and `<xml-tag>` are all invalid names.

Closing tags

Every XML element must have an associated closing tag to be valid. Empty elements that contain no data can use a closing tag or the shorthand method that puts a “/” at the end of the opening tag. Regular and empty XML elements might look like this:

```
<mytag> data </mytag>
<emptytag/>
```

Empty tags are usually only included in a document for the value of an attribute that they contain.

Attributes

XML elements can contain one or more attributes which define values that are relevant to that element. Each attribute has a name, following the same naming conventions as the element name, and a value assigned with the “=” operator. Unlike HTML, all attribute values must be enclosed in quotation marks to be valid.

```
<security level= "classified"> Confidential </security>
```



being well-formed.

Omission of closing tags is an error that prevents the XML document



are recommended.

Either single or double quotes can be used for attribute values but double quotes

The XML declaration

The start of the very first line of every XML document should contain the XML declaration. This identifies the document to be a XML document and defines the version of XML being used.

The XML declaration is stated in a special tag that starts “<?” and ends “?>” to denote that it contains a processing instruction. This type of tag can also be used to state other processing instructions, such as specifying the stylesheet that is to be used with that XML document.

Inside the tag, the element is named “xml” to denote that it is part of the XML specification itself. To define the XML version being used, the tag adds an attribute named “version” that is assigned the version number as its value. Currently the version is 1.0 so the XML declaration looks like this:

```
<?xml version="1.0"?>
```



All attribute values must be enclosed by quotation marks.

The XML declaration can also contain additional attributes to define other information about the document such as character encoding or document dependency.

Character encoding in XML follows the Unicode standard which means that the XML processor will recognize UTF-8 encoded documents, such as those written in English, and also UTF-16 encoded documents, such as those written in Japanese. Most XML processors also support the Latin-1 encoding (ISO-8859-1), that is used by Windows, but encodings other than UTF-8 or UTF-16 can be specified in the XML declaration like this example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

The XML declaration may include an attribute called “standalone” to specify if that document uses other documents. This attribute can only have a value of either “yes” or “no”. Most XML documents use other files, such as a stylesheet and schema, but completely independent XML documents might start with this declaration:

```
<?xml version="1.0" standalone="yes"?>
```

Elements

Every valid XML document must always have one single element that entirely encloses all the other elements in the document. This element is called the “root” element.

Elements that are directly contained within the root element are said to be “nested” within the root element. Further elements can be nested within those elements to form a sort of hierarchy.

An element that is enclosed by another element is called a “child” of the containing element. Conversely the containing element is called the “parent” of the enclosed element.

In the code shown below the root element is named “personnel”. The personnel root element contains two child elements that are each named “entry”. Each entry element encloses its own child element called “name” that contains some content data.

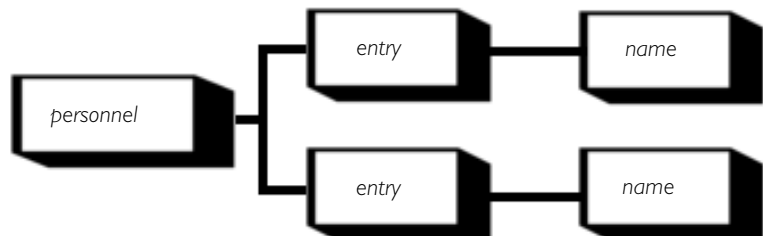


If the elements are not correctly nested the XML document will not be well-formed.

```
<?xml version="1.0" ?>  
<personnel>  
  <entry>  
    <name>John Smith</name>  
  </entry>  
  <entry>  
    <name>Sally Jones</name>  
  </entry>  
</personnel>
```

It is important to note that all elements must be correctly nested by writing their closing inner tags before writing the closing tag for their outer parent element.

The hierarchical nature of the elements in a XML document can be seen as a tree of elements, with the root element as its base.



Element attributes

One, or more, attributes can be added to an element to store additional information about the element's actual content. This information about other data is called "meta-information".

Each attribute must have a name and a value.

The attribute value is assigned to the attribute name using the "=" operator and the value must be enclosed in quotes. If the assigned value itself contains a quoted string the type of quotation marks must differ from those used to enclose the entire value. For instance, if double quotes are used to enclose the whole value then use single quotes for the string like this:

```
<name familiar="'Jack'">John Smith</name>
```



When naming attributes, you must follow the same naming conventions that

are used when naming elements – see page 21.

The attributes in an element must each have a unique name to avoid confusing the XML parser. Also the value of an attribute cannot contain a "<" character, but this can be represented with the entity reference "<" instead.

An alternative way to attach meta-information to element data is to use nested elements. The attribute example shown above is included in the code below using this alternative method:

```
<?xml version="1.0" ?>
<personnel>
  <entry>
    <name>John Smith</name>
    <familiar>'Jack'</familiar>
  </entry>
  <entry>
    <name>Sally Jones</name>
    <tel>555-1234</tel>
    <tel preferred="true">555-8282</tel>
  </entry>
</personnel>
```

Attributes are generally less suited to containing explicit data but more typically are useful to supply information about the status or preference of an element's actual content. The code above illustrates this when stating a preferred contact telephone number.

Empty elements

Elements that do not enclose any child elements or textual data are called “empty elements”. Although they have no content that can be written out as text, empty elements are useful to include information inside their attributes. For instance, you might want to include information regarding the source of an image file that is relevant to a particular entry.

Empty elements can have a normal closing tag, or can use the shorthand method that combines both tags by adding a closing slash at the end of the opening tag.

The code below demonstrates both methods being used to indicate the source of relevant image files in the element called “photo”:



Attribute values must always be enclosed in quotes.

```
<?xml version="1.0" ?>
<personnel>
  <entry>
    <name>John Smith</name>
    <familiar>'Jack'</familiar>
    <photo filename="jsmith.jpg"></photo>
    <tel>555-6363</tel>
  </entry>
  <entry>
    <name>Sally Jones</name>
    <familiar>'Sal'</familiar>
    <photo filename="sjones.jpg" />
    <tel>555-1234</tel>
    <tel preferred="true">555-8282</tel>
  </entry>
</personnel>
```

It does not matter whether empty elements are closed with the shorthand method or with a separate closing tag as XML regards both methods as equally correct.

Empty elements, or any other elements, that are not closed will mean that the XML document is not well-formed and will cause an error to be generated by the XML parser.

Comments

It is always good practice to add comments to any source code to annotate sections of the code so that their purpose is clear when read by a third party, or when the source code is revisited after a period of time.

XML uses exactly the same syntax as HTML for comments so that any text that is inserted between “<!--” and “-->” will be ignored by the XML parser.

A double hyphen may not be used inside comment tags as this will confuse the parser. This means that a comment may not be nested inside another comment. Also note that comments should not be added inside an element tag but must appear either before or after a complete tag.

It is sometimes useful to temporarily add comment tags around a section of XML code to hide it from the parser during development or debugging. This is called “commenting-out” and can be used to isolate a piece of code that is creating an error so that it can be identified and rectified.

The comment text may be just a few words or many lines but the entire comment will be invisible to both the XML parser and the web browser. The example below includes several comments:



Add a date comment to each XML document to establish when

the code was created or when it was last updated.

```
<?xml version="1.0" ?>
<!-- staff list, last updated April 5, 2003 -->
<personnel>
<!-- staff member -->
  <entry>
    <!-- full name -->
    <name>John Smith</name>
    <!-- commonly known as -->
    <familiar>'Jack'</familiar>
    <!-- mugshot -->
    <photo filename="jsmith.jpg"></photo>
    <!-- home telephone number -->
    <tel>555-6363</tel>
  </entry>
</personnel>
```

Character entities

The XML specification predefines five special pieces of code for characters that are normally recognized as part of the XML language itself. These are called “character entities” and they must be used if you want to include those characters as part of an element’s content. Character entities use a special syntax to represent the character in a way that avoids the XML parser interpreting the character as code.

The standard XML character entities are as follows:

- `<`; represents the “<” left angle bracket.
- `>`; represents the “>” right angle bracket.
- `&`; represents the “&” ampersand character.
- `'`; represents the “'” single quote apostrophe.
- `"`; represents the “” double quotation mark.

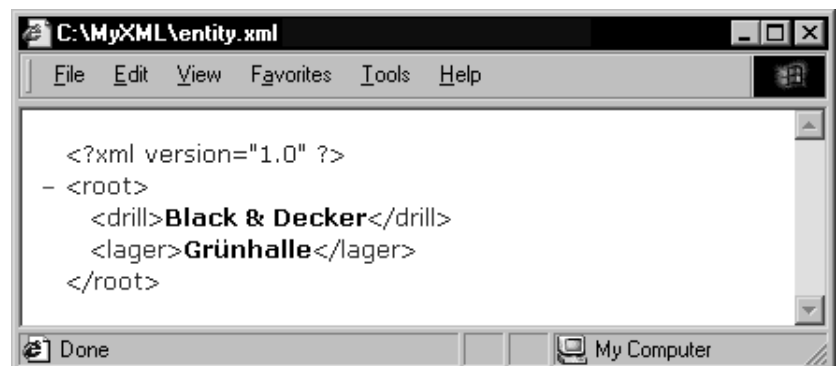
XML also supports character references that create a character using its Unicode character code. These are written in the XML document as “`&#nnn;`” where “nnn” is the reference number. This is useful to add characters that are not available on your keyboard.



The Windows Character Map application can be used to find the Unicode

reference number of many characters.

```
<?xml version="1.0"?>
<root>
  <drill>Black &amp; Decker</drill>
  <lager>Gr&#252;nhalle</lager>
</root>
```



CDATA blocks

Character data that is not required to be interpreted by the XML parser can be “escaped” inside a CDATA block of code in a XML document.

The CDATA blocks are enclosed by “<![CDATA[” and “]]>” so that anything between will be preserved literally in the document. This is useful to incorporate examples of markup without having to escape all the recognized markup characters individually.

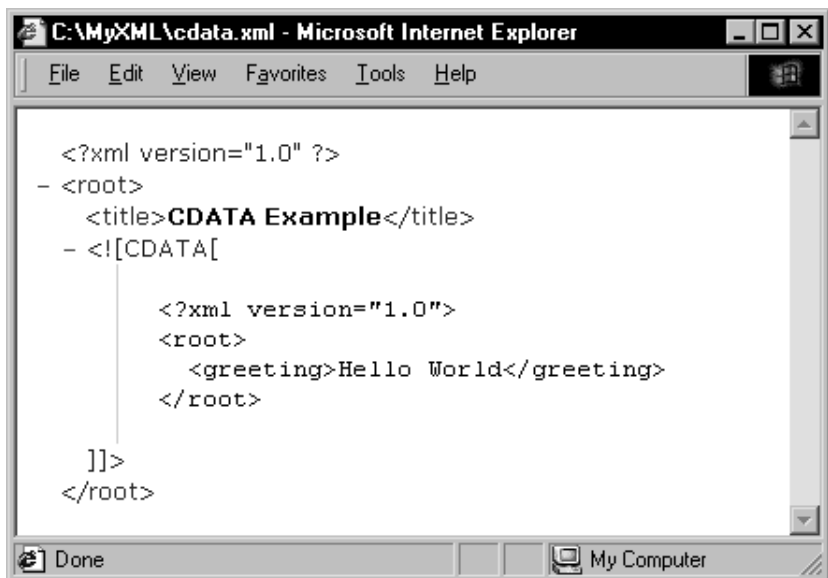
In the code below a CDATA block includes an entire XML document that is preserved when viewed in Internet Explorer:



A XML document can have multiple CDATA blocks but they cannot be nested inside

each other.

```
<?xml version="1.0"?>
<root>
  <title>CDATA Example</title>
  <![CDATA[
    <?xml version="1.0">
    <root>
      <greeting>Hello World</greeting>
    </root>
  ]]>
</root>
```



XML example document

The XML basics that are demonstrated in this chapter are evidence that XML is itself very simple. The power of XML lies in what can be achieved with the data that is described in XML.

To summarize XML syntax, the code below incorporates some of the features of XML in a single document:



personnel.xml



The XML parser ignores indents and line breaks so XML code is normally formatted to make it more easily human-readable.

```
<?xml version="1.0" standalone="no" ?>
<?xml:stylesheet href="personnel.css" type="text/css" ?>
<!-- management list, last updated April 5, 2003 -->
<personnel>
  <heading>Company Managers</heading>
  <entry>
    <job>Sales & Marketing Manager</job>
    <name>John Smith</name>
    <tel preferred="true">555-9494</tel>
    <tel>555-6363</tel>
  </entry>
  <entry>
    <job>Accounts Manager</job>
    <name>Sally Jones</name>
    <tel>555-8282</tel>
    <photo filename="sjones.jpg" />
  </entry>
  <entry>
    <job>Production Manager</job>
    <name>Bob Solomon</name>
    <tel>555-7171</tel>
  </entry>
</personnel>
```

The stylesheet referred to in this document is shown on the next page, together with an illustration of how this XML data may be displayed by the stylesheet in a web browser.