How-To Guide for Learning Basic XML

By Julie Odenbach

XML stands for Extensible Markup Language, and it provides us with a convenient way to define and store information so we can then share that information with different computers, websites, and applications.

An XML file or document builds a tree structure. The root element serves as the parent element, and that parent element contains its child elements. Each child element may also include one or more child elements. One significant advantage of XML is that it is easily readable to humans and easy to parse with software.

XML stores data in plain text without specifying how that data should be displayed. Use HTML to display the information on web pages in an easy-to-read manner. Remember that HTML is case-insensitive, while XML is case-sensitive.

XML Root Element

All XML starts with a **root element**, and all other elements must be correctly placed within that parent root element. In the example below, “WeatherReport” is our root element. We open it with our “<WeatherReport>” tag. Each element must be closed using the element closing tag, such as “</WeatherReport>”.

```
<WeatherReport>                    ➔ This is the root element
        … child tags will go here …
</Weather Report>                  ➔ This closes the root element
```

XML Tags

Within our WeatherReport XML document, we can define additional **child data elements**. Let’s add more information to our XML example:

```
<xml version=”1.0”?>
<WeatherReport>
        <Date>Thursday  06/15/2023</Date>
        <TimeOfDay>3:52 PM</TimeOfDay>
        <TempF>84</TempF>
        <WindDirection>South21west</WindDirection>
        <WindSpeed>21 mph</WindSpeed>
        <DewPointF>52</DewPoint>
        <BarometricPressure>1015.5<BarometricPressure>
</WeatherReport>
```

We have added more detail about what information can and should be supplied with our XML document to create a valid WeatherReport.

All XML elements must have a closing tag. So, a typical XML tag would look like ***"<tag> … </tag>.***

XML Attributes and Elements

XML elements and attributes can be used interchangeably. Both forms are allowed. Attributes specify a value for one specific field, such as ***<Meeting date="06/15/2023>"***, while an element can specify one or more values for a field, such as "***<Date>06/15/2023</Date>".***

Since attributes can't contain multiple values, elements may offer more flexibility in defining your data structure. In some cases, however, people may wish to provide a unique id attribute to identify different sections in their XML. In that case, it's recommended that metadata (data about the data) be defined as an attribute, while the data itself is better represented as elements.

We can store data for our meeting notice using **elements**:

```
<?xml version="1.0"?>
<Meeting>
        <Date>06/15/2023</Date>
        <MeetingTitle>Resolve database issues</MeetingTitle>
        <StartTime>4:00 PM Central</StartTime>
        <EndTime>4:30 PM Cerntral</EndTime>
        <MeetingAttendee>mandy.shaw@gmail.com</MeetingAttendee>
        <MeetingAttendee>tim.lawrence@gmail.com</MeetingAttendee>
</Meeting>
```

You can also see that we have included two meeting attendees using the same tag for each. This is allowed, and each of those multiple elements can be parsed and retrieved.

Or we can store the same data using an **attribute**:

```
<?xml version="1.0"?>
<Meeting date="06/15/2023">
        <MeetingTitle>Resolve database issues</MeetingTitle>
        <StartTime>4:00 PM Central</StartTime>
        <EndTime>4:30 PM Central</EndTime>
        <MeetingAttendee>mandy.shaw@gmail.com</MeetingAttendee>
        <MeetingAttendee>tim.lawrence@gmail.com</MeetingAttendee>
</Meeting>
```

Note that attribute values must always be quoted, as **date="06/15/2023"** Is quoted in this example.

XML DTD

DTD stands for Document Type Definition, and it defines the legally allowed elements and attributes of your XML document.

An XML document using correct syntax is said to be “Well Formed”, while an XML document validated against a DTD is said to be both “Well Formed” and “Valid”.

This example of a DTD file defines the structure and elements of an XML document, as shown below. This XML document’s root element is defined as “TeamMember”, and the document can contain definitions for the address, name, email, and phone elements. The name field may contain entries for both the first and last names. The two data types are “PCDATA” and CDATA. PCDATA is parseable character data, while CDATA is not normally parsed.

```
<!DOCTYPE TeamMember
[
<!ELEMENT address (name, email, phone)>
<!ELEMENT name (first, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
]>
```

You do not need to use a DTD to define your XML. The advantage of using a DTD for exchanging XML information is that you can verify your own XML and any incoming XML from an external source.

Your DTD file can be provided within the same file as your XML as long as it precedes your XML. You can also specify your DTD file by giving the name of your DTD file in your XML document.

```
<?xml version="1.0"?>
<!DOCTYPE address SYSTEM "TeamMember.dtd">
<TeamMember>
  <name>
    <first>Roxanne</first>
    <last>Jensen</last>
  </name>
  <email>roxanne.jensen@gmail.com</email>
  <phone>701-555-4612</phone>
</TeamMember>
```

Your third option is to specify an external DTD file with a URL, as shown below:

```
<!DOCTYPE Catalog PUBLIC "xyzCompany/Catalog""http://abc.xyzCompany.org/dtds/TeamMember.dtd">
```

.

XML Version

An XML version of “1.0” means this XML file uses encoding=”utf-8” Unicode encoding. You could also explicitly specify <?xml version=”1/0” encoding=”utf-8”>. The default encoding is UTF-8 or UTF-16, so you would not need to specify the version if that default encoding works for your XML document.

Summary

In summary, defining XML is a great way to define the information required to meet your goals. For more detailed information on XML and DTD files, check out the **W3Schools website** at *https://www.w3schools.com/xml/xml_dtd.asp. You can learn how to specify which* tags need to be supplied, if those tags are required or may be empty, how many values may be supplied for those tags, and more. Defining XML is always fun and provides a great way define the rules for your project’s data.