e-PG Pathshala

Subject: Computer Science

Paper: Web Technology

Module: XML DTD

Module No: CS/WT/8

Quadrant 1 – e-text

Learning Objectives

The last module gave an introduction about XML, the document structure and the XML rules for creating a valid XML document. In this module we will learn about XML elements and attributes in detail. This module also provides us an understanding about XML Document Type Definitions (DTD) by learning about the basics of creating an XML document and its DTD.

Document Type Definitions

A DTD (Document Type Definition) describes the structure of one or more XML documents. Specifically, a DTD describes about Elements, Attributes and Entities defined in an XML document. An XML document is called a well-structured XML document if it follows certain simple syntactic rules and an XML document is called a valid XML document if it also specifies and conforms to a DTD.

The need for DTDs

XML documents are designed to be processed by computer programs and if we can put just *any* tags in an XML document, it's very hard to write a program that knows how to process the tags. A DTD specifies what tags may occur, when they may occur, and what attributes they may or must have. A DTD allows the XML document to be verified (shown to be legal) by the browser. A DTD that is shared across groups allows the groups to produce consistent XML documents.

An XML example

Following is an simple example of an XML document.

```
<novel>
   <foreword>
      <paragraph>This is the great Indian novel.</ paragraph>
   </foreword>
   <chapter number="1">
      <paragraph>It was a dark and stormy night.</paragraph>
      <paragraph>Suddenly, a shot rang out!</paragraph>
   </chapter>
</novel>
```

This XML document contains elements and attributes and the DTD describes about the elements, such as <novel>, <foreword>, <chapter> and <paragraph>, consisting of *tags* and *content*. The element <novel> consists of elements <foreword> and <chapter>. The element <chapter> consists of attributes, such as number="1", consisting of a *name* and a *value*.

This XML file is described with the following DTD,

A DTD example

```
<!DOCTYPE novel [
  <!ELEMENT novel (foreword, chapter+)>
  <!ELEMENT foreword (paragraph+)>
  <!ELEMENT chapter (paragraph+)>
  <!ELEMENT paragraph (#PCDATA)>
  <!ATTLIST chapter number CDATA #REQUIRED>
]>
```

The DTD of an XML document starts with <!DOCTYPE root-element […]>

In this example we have defined as,

```
<!DOCTYPE novel [ …]>
```

In an DTD the elements are described as<!ELEMENT…> and attributes of the element are described as <!ATTLIST ….>.

Here in this example, a novel consists of a foreword and one or more chapters, in that order. A foreword consists of one or more paragraphs. A chapter also consists of one or more paragraphs. A paragraph consists of parsed character data which means the text that cannot contain any other elements.

Each chapter must have a number attribute given as “REQUIRED”.

ELEMENT descriptions

An element content can be another element or a list of child elements or it can be a text string or numeric value (character data).

There are suffixes added to the element content where an element consists of zero, one or more occurences of the child elements. A (?) specifies optional or zero, (+) denotes one or more elements can occur and a (*) denotes zero or more of the elements can occur.

Suffixes:

?	optional	foreword?
+	one or more	chapter+
*	zero or more	appendix*

Seperators can also be added with the element declaration, that specifies the order of occurrence when the element has multiple number of child elements. There are two ways we can use to denote the order of occurrence. Using a comma (,) between the child elements specifies that the child elements will occur in the order given. Using a vertical line (|) means that either one or the other child element will occur.

Separators:

,	both, in order	foreword?, chapter+
|	or	section|chapter

Grouping:

The elements can be grouped in the element content with parenthesis which can also be specified along with the order of occurrence or the number of occurrences of the child elements.

( ) grouping (section|chapter)+

Elements without children

Elements can be defined as empty or without children.

<!ELEMENT name category>

The *name* is the element name used in start and end tags and the *category* may be given as EMPTY.

In the DTD the
 tag can be defined as : <!ELEMENT br EMPTY>

In the XML the tag can be used as:
</br> or just

In the XML, an empty element may not have any content between the start tag and the end tag but an empty element may (and usually does) have attributes.

Elements with unstructured children

The elements can be defined with unstructured children.

```
<!ELEMENT name category>
```

Here the category may be ANY.

This indicates that *any* content character data, elements, even undeclared elements may be used. Since the whole point of using a DTD is to define the structure of a document, ANY should be avoided wherever possible.

The category may be (#PCDATA), indicating that only character data may be used. Here the parentheses are required.

In the DTD when an element is defined as (#PCDATA), the syntax is given as,

```
<!ELEMENT paragraph (#PCDATA)>
```

In the XML file the element can contain text content

```
<paragraph>A shot rang out!</paragraph>
```

Note: In (#PCDATA), whitespace is kept exactly as entered. Elements may *not* be used within parsed character data. Entities are character data, and *may* be used.

Elements with children

An element can contain one or more children.

A category may describe one or more children as given in the syntax below,

```
<!ELEMENT novel (foreword, chapter+)>
```

Here the parentheses are required, even if there is only one child, a space must precede the opening parenthesis. Commas (,) between elements mean that *all* children must appear, and must be *in the order specified*. The "|" separators means any one child may be used. All child elements must themselves be declared. Children may have children and parentheses can be used for grouping.

An complete example of the syntax we have seen so far is given in the following example.

```
<!ELEMENT novel (foreword, (chapter+|section+))>
```

Elements with mixed content

Elements can also contain mixed content with text data or elements or both.

Here #PCDATA describes elements with only character data and #PCDATA can be used in an "or" grouping also as follows,

```
<!ELEMENT note (#PCDATA|message)*>
```

This is called a mixed content

But certain and rather severe restrictions apply:

- #PCDATA must be first
- The separators must be "|"
- The group must be starred (meaning zero or more)

Names and namespaces

All names of elements, attributes, and entities, in both the DTD and the XML, are formed by following the rules given as below,

- The name must begin with a letter or underscore.
- The name may contain only letters, digits, dots, hyphens, underscores, and colons (and, for foreign languages, combining characters and extenders).

The DTD doesn't know about namespaces as far as it knows, a colon is just part of a name

The following are different (and both legal),

```
<!ELEMENT chapter (paragraph+)>
<!ELEMENT myBook:chapter (myBook:paragraph+)>
```

We need to avoid colons in names, except to indicate namespaces.

An expanded DTD example

An example covering all those concepts we have seen so far is given now.

```
<!DOCTYPE novel [
  <!ELEMENT novel
     (foreword, chapter+, biography?, criticalEssay*)>
  <!ELEMENT foreword (paragraph+)>
  <!ELEMENT chapter (section+|paragraph+)>
  <!ELEMENT section (paragraph+)>
  <!ELEMENT biography(paragraph+)>
  <!ELEMENT criticalEssay (section+)>
```

```
    <!ELEMENT paragraph (#PCDATA)>
]>
```

The root element is the <novel> element. It consists of elements such as <foreword>, one or more <chapter> elements, zero or one <biography> elements and zero or more <criticalEssay> elements. The <foreword> elements consists of one or more <paragraph> elements. The <chapter> elements consists of one or more <section> or <paragraph> elements.The <section> element consists of one or more <paragraph> elements. The element <biography> consists of one or more <paragraph> elements. The element <criticalEssay> consists of one or more <section> elements. The element <paragraph> consists of #PCDATA.

Attributes and entities

In addition to elements, a DTD may declare attributes and entities. An attribute describes information that can be put within the start tag of an element.

In the XML file we define the attributes as follows,

```
<dog name="Spot" age="3"></dog>
```

In an DTD file the attribute is described as,

```
<!ATTLIST dog
        name  CDATA  #REQUIRED
        age    CDATA  #IMPLIED >
```

An entity describes text to be substituted.

In an XML file an entity is defined as,

```
&copyright;
```

In the DTD it is described as,

```
<!ENTITY copyright "Copyright Dr. Dave">
```

Attributes

The format of defining an attribute is,

```
<!ATTLIST element-name
        name  type  requirement
        name  type  requirement>
```

where the *name-type-requirement* may be repeated as many times as desired.

Note that only spaces separate the parts, so careful counting is essential. The *element-name* tells which element may have these attributes.

The *name* is the name of the attribute. Each element has a *type*, such as CDATA (character data)

The *requirement* may be defined as required, optional, or "fixed". In the XML, attributes may occur in any order

Important attribute types

There are ten attribute types and these are the most important ones:

- CDATA — The value is character data
- (man|woman|child) — The value is one from this list
- ID — The value is a unique identifier. *Note:* ID values must be legal XML names and must be unique within the document
- NMTOKEN — The value is a legal XML name. This is sometimes used to disallow whitespace in the name. It also disallows numbers, since an XML name cannot begin with a digit.

Less important attribute types,

- IDREF — The ID of another element
- IDREFS — A list of other IDs
- NMTOKENS — A list of valid XML names
- ENTITY — An entity
- ENTITIES — A list of entities
- NOTATION — A notation
- xml: — A predefined XML value

Requirements

Recall that an attribute has the form,

<!ATTLIST *element-name name type requirement*>

The *requirement* is one of a default value, enclosed in quotes,

Example: <!ATTLIST degree CDATA "PhD">

The *requirement* can take different values such as #REQUIRED or #IMPLIED or #FIXED.

- #REQUIRED means the attribute must be present.
- #IMPLIED means the attribute is optional.
- #FIXED "value" the attribute always has the given value. If specified in the XML, the same value must be used.

Entities

Entities are Variables used to define shortcuts to common text. Entities should not be confused with character references, which are numerical values between & and #.

There are exactly five predefined entities: <, >, &, ", and '.

Additional entities can be defined in the DTD by the following syntax,

```
<!ENTITY copyright "Copyright Dr. Dave">
```

Entities can be defined in another document by,

```
<!ENTITY copyright SYSTEM "MyURI">
```

An example of use in the XML as,

This document is &copyright; 2002.

Another example: XML

```
<?xml version="1.0"?>
<!DOCTYPE weatherReport SYSTEM "http://www.mysite.com/mydoc.dtd">
<weatherReport>
  <date>05/29/2002</date>
  <location>
    <city>Chennai</city> <state>Tamilnadu</state>
    <country>INDIA</country>
  </location>
  <temperature-range>
    <high scale="F">84</high>
    <low scale="F">51</low>
  </temperature-range>
</weatherReport>
```

The DTD for this example

```
<!ELEMENT weatherReport (date, location, temperature-range)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT location (city, state, country)>
<!ELEMENT city (#PCDATA)>
```

```
<!ELEMENT state (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT temperature-range ((low, high)|(high, low))>
<!ELEMENT low (#PCDATA)>
<!ELEMENT high (#PCDATA)>
<!ATTLIST low scale (C|F) #REQUIRED>
<!ATTLIST high scale (C|F) #REQUIRED>
```

Inline DTDs

If a DTD is used only by a single XML document, it can be put directly in that document. An inline DTD can be used only by the document in which it occurs.

```
<?xml version="1.0">
<!DOCTYPE myRootElement [
   <!-- DTD content goes here -->
]>

<myRootElement>
    <!-- XML content goes here -->
</myRootElement>
```

Inline DTDs - Example

```
<!DOCTYPE company [
   <! ELEMENT company    ( (person | product)*) >
   <! ELEMENT person      ( ssn, name, office, phone?) >
   <! ELEMENT ssn         ( # PCDATA) >
   <! ELEMENT name        ( # PCDATA) >
   <! ELEMENT office      ( # PCDATA) >
   <! ELEMENT phone       ( # PCDATA) >
   <! ELEMENT product     ( pid, name, description? ) >
   <! ELEMENT pid         ( # PCDATA) >
   <! ELEMENT description  ( # PCDATA) >
]>

<company>
<person> <ssn> 12345678 < /ssn>
          <name> John </name>
          <office> B432 </office>
          <phone> 1234 </phone>
</person>
<product> ... </product>
...
</company>
```

External DTDs

An external DTD which is a DTD defined as a separate document is declared with a SYSTEM or a PUBLIC command. The syntax is given as follows,

```
<!DOCTYPE myRootElement SYSTEM "http://www.mysite.com/mydoc.dtd">
```

The name that appears after DOCTYPE (in this example, myRootElement) must match the name of the XML document's root element. We will use SYSTEM for external DTDs that we define for ourself and we will use PUBLIC for official and published DTD's. External DTDs can only be referenced with a URL.

The file extension for an external DTD is .dtd.

External DTD's are almost always preferable to inline DTD's, since they can be used by more than one document.

External DTDs - Example

```
<!DOCTYPE company SYSTEM "company.dtd">
<company>
<person> <ssn> 12345678 < /ssn>
          <name> John </name>
          <office> B432 </office>
          <phone> 1234 </phone>
</person>
<product> ... </product>
...
</company>
```

In this example the DTD is defined in a separate document called "company.dtd" and referred in the XML document after DOCTYPE.

The DTD can reside at a different URL, and then we refer to it as,

```
<!DOCTYPE company SYSTEM "http://www.mydtd.com/company.dtd">
```

An example of **TV Schedule DTD** is given,

```
<!DOCTYPE TVSCHEDULE [
<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER, DAY+)>
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY ((DATE, HOLIDAY) | (DATE, PROGRAMSLOT+))+>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME, TITLE, DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATING CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
```

Another Example for **Newspaper Article DTD** is also given for understanding,

```
<!DOCTYPE NEWSPAPER [
<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE, BYLINE, LEAD, BODY, NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>
<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>
<!ENTITY NEWSPAPER "Vervet Logic Times">
<!ENTITY PUBLISHER "Vervet Logic Press">
<!ENTITY COPYRIGHT "Copyright 1998 Vervet Logic Press">
]>
```

Limitations of DTDs

DTDs are a very weak specification language and we can't put any restrictions on element contents. It is difficult to specify when all the children must occur, but may be in any order or this element must occur a certain number of times. There are only ten data types for attribute values.

But most of all DTDs aren't written in XML. If we want to do any validation,we need one parser for the XML *and* another for the DTD. This makes XML parsing harder than it needs to be. In order to support this, there is a newer and more powerful technology called XML Schemas. However, DTDs are still very much in use.

Summary

This module provides knowledge about XML DTD.The module identifies the purpose of DTD with few examples. The module also discusses about Inline and External DTD's.