DTD Basics

Chapter 03 to 04

Topics

- What is DTD?
- Valid documents
- Internal and External DTD
- How to write DTD
- Associating DTD with XML documents
- DOCTYPE declaration
- Validation and non-validating parser
- Working with element and element content model
  - → PCDATA
  - → MIXED
  - → ANY
  - → Sequence
  - → Choice

We are going to learn how to write DTD rules. After writing some DTD rules we should write xml and validate the xml against the DTD. For that we need a DTD validator or a parser that can validate xml against DTD. Visual Studio can validate DTD and we can use it easily. Third-party free tools are also available. We are using Internet Explorer for parsing XML. IE uses MSXML parser. IE does not validate XML by default. But using some plug-in we can use IE to validate XML with DTD. Collect the plug-in. This plug-in works fine with IE version up to 9.

What is DTD?

- DTD Stands for Document Type Definition
- DTD specifies the grammar rules for the particular type of XML document.
- What elements an XML document should contain and what contents the elements should contain are defined using DTD.
- By associating a DTD rule with XML documents you can make sure, XML documents are written conforming to certain standards.
- A DTD defines allowable elements, attributes and their content model for a certain class of XML document.

Well-formed vs. Valid XML document

- A well-formed XML written with valid XML syntaxes and its elements properly nested.
- A valid XML is written following the rules defined in a DTD for the documents and its elements, attributes and content model matches the DTD definitions
- A well-formed XML document is not always valid but a valid XML document is always well-formed.

How to write DTD?

- DTD definitions or reference are defined just after the XML declaration and before the root element
- DTD definitions or reference is declared in DOCTYPE element
- DTD can be internal or external
- Internal DTD rules are defined in the XML document itself

```
<? xml version='1.0' ?>
<!DOCTYPE documentelement [.. DTD definitions..]>
<documentelement>
        .
        .
</documentelement>
```

- External DTD are defined in a separate file and that DTD file is referenced in XML document

```
<?xml version='1.0'?>
<!DOCTYPE documentelement … external DTD reference..>
<documentelement>
.
.
</documentelement>
```

- documentelement is the root element. The element name just after the DOCTYPE must match the root element

DOCTYPE declaration

- The document type (DOCTYPE) declaration consists of an internal, or references an external Document Type Definition (DTD).
- A document can use both internal and external DTD subsets.
- The internal DTD subset is specified between the square brackets of the DOCTYPE declaration.
- The declaration for the external DTD subset is placed before the square brackets immediately after the SYSTEM keyword.
- Rules
  - → The document type declaration must be placed between the XML declaration and the first element (root element) in the document.
  - → The keyword DOCTYPE must be followed by the name of the root element in the XML document.
  - → The keyword DOCTYPE must be in upper

Things to know about Parsers

- Parsers reads the xml document and realize the structure of the document and breaks it down into form that you can manipulate it in an application program
- Parsers generally transform the xml into a tree structure inside the computer RAM
- There two types of parsers
  - → Non-validating parsers - only checks well-formness of XML
  - → Validating parsers - it checks well-formness and it also validates the xml against some DTD rules or Schema definition

Internal DTD

- The example below (ex01.xml) show XML with internal DTD rules

```
<?xml version='1.0'?>
<!DOCTYPE contact [
      <!ELEMENT contact (#PCDATA)>
      ]>
<contact>907865</contact>
```

  - → The DTD declaration says that the document is about contact (root element or document element)
  - → Inside definition, it declares an element contact whose content model is declared inside parenthesis ()
  - → contact can contain PCDATA (Parsed character data)
  - → Parsed Character Data are simple text that is parsed by the parser and should not contain illegal character like <, & etc.

External DTD

- External DTDs are useful for creating a common DTD that can be shared between multiple documents.
- Any changes that are made to the external DTD automatically updates all the documents that reference it.
- Example
- The DTD rules defined a separate file (ex02.dtd)
- <!ELEMENT contact (#PCDATA)>
- The DTD file is referenced in the XML document(ex02.xml)

```
<?xml version='1.0'?>
<!DOCTYPE contact SYSTEM "contact.dtd">
<contact>907865</contact>
```

- NOTICE use of SYSTEM, it is called SYSTEM identifier
- External DTD can be referenced using

1. SYSTEM identifiers: it is used to reference a DTD file on your local computer
2. PUBLIC identifier : It uses a specialized identifier to locate actual DTD file

Working with Element content model

- An XML Element can contain of type

→ Parsed Character Data
→ Any
→ Empty
→ Mixed
→ Other elements and that can have
  - Sequence
  - Choice
  - Cardinality

Working with PCDATA

- PCDATA stands for parsed character data, that is, text that is not markup.
- Therefore, an element that has the allowable content (#PCDATA) may not contain any children.
- But PCDATA accepts empty data
- PCDATA does not allow markup characters like &, < etc.
- Example
- The example (ex03.dtd) defines the content for doc-content element

```
<!ELEMENT doc-content (#PCDATA)>
```

- The valid XML document (ex03-valid.xml) according to ex03.dtd

```
<?xml version='1.0'?>
<!DOCTYPE doc-content SYSTEM "ex03.dtd">
<doc-content>
Some content &amp;amp; more
</doc-content>
```

- & is not valid in PCDATA, it is written as entity-reference
- If you change the ex03-valid.xml like below, the xml is also valid

```
<?xml version='1.0'?>
<!DOCTYPE doc-content SYSTEM "ex03.dtd">
<doc-content>
</doc-content>
```

- But the following ex03-invalid.xml is not valid

```
<?xml version='1.0'?>
<!DOCTYPE doc-content SYSTEM "ex03.dtd">
<doc-content>
<text>Some content &amp;amp; more</text>
</doc-content>
```

ANY content

- Refers to anything at all, as long as XML rules are followed.
- It can contain text, other elements but the elements must be declared
- The example (ex04.dtd) defines any content for doc-content element

```
<!ELEMENT doc-content ANY>
<!ELEMENT b (#PCDATA)>
```

- The xml file (ex04-valid.xml) is valid

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE doc-content SYSTEM "ex04.dtd">
<doc-content>
      <b>DTD</b> defines content-model
</doc-content>
```

- But the following (ex04-invalid.xml) is invalid because element i is not declared

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE doc-content SYSTEM " ex04.dtd">
<doc-content>
  <b>DTD</b> defines <i>content-model</i>
</doc-content>
```

Child elements

- You can place any number of element types within another element type. These are called children elements, and the elements they are placed in are called parent elements.
- Children element types are declared using parentheses in the parent element type's declaration
- The child element must be declared in a separate element type declaration.
- The example (ex05.dtd) show child element declaration

```
<!ELEMENT doc (doc-content) >
<!ELEMENT doc-content (#PCDATA)>
```

- The xml (ex05.xml) according the DTD rules

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE doc SYSTEM "ex05.dtd">
<doc>
  <doc-content>
    Some text content
  </doc-content>
</doc>
```

Children sequence

- Multiple children are declared using commas (,). Commas fix the sequence in which the children are allowed to appear in the XML document.
- The example (ex06.dtd) shows sequence declaration

```
<!ELEMENT doc (title, doc-content) >
<!ELEMENT title (#PCDATA)>
<!ELEMENT doc-content (#PCDATA)>
```

- The following (ex06-valid.xml) is a valid document

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE doc SYSTEM "ex06.dtd">
<doc>
  <title>DTD rules</title>
  <doc-content>
    Some text content
  </doc-content>
</doc>
```

- But the following (ex06-invalid.xml) is invalid because it breaks the sequence – doc-content is before the title element

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE doc SYSTEM "ex06.dtd">
<doc>
  <doc-content>
    Some text content
  </doc-content>
  <title>DTD rules</title>
</doc>
```

Child cardinality or multiplicity

- Cardinality specifies how many times a child element can appear.
- Child cardinality is specified using symbols.
- The following cardinality symbols and their meaning

Symbol	Meaning
Nil	Element is required and it should appear only once
*	Element can appear zero or many times
+	Element can appear 1 or many times
?	Element can appear zero or 1 time

- The example (ex09.dtd) use cardinality of children

```
<!ELEMENT docs (doc*)>
<!ELEMENT doc (title, doc-content, summary?) >
<!ELEMENT title (#PCDATA)>
<!ELEMENT doc-content (para+)>
<!ELEMENT para (#PCDATA)>
<!ELEMENT summary (#PCDATA)>
```

- docs may contain 0 or many doc elements
- doc must contain title, doc-content and optionally can contain summary once
- do-content can contain many para element but at least one

- The xml file (ex09.xml) is valid according to dtd rules

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE docs SYSTEM "ex09.dtd">
<docs>
 <doc>
  <title>DTD rules</title>
  <doc-content>
   <para>
    Some text content
   </para>
   <para>
    Some more text content
   </para>
  </doc-content>
 </doc>
 <doc>
  <title>XSD rules</title>
  <doc-content>
   <para>
    Some text content
   </para>
   <para>
    Some more text content
   </para>
  </doc-content>
  <summary>Summary content</summary>
 </doc>
</docs>
```

MIXED content

- Mixed content is used to declare elements that contain a mixture of children elements and text (PCDATA).
- The (#PCDATA) and children element declarations must be separated by the (|) operator.
- (#PCDATA) must come first in the mixed content declaration.
- The operator (*) must follow the mixed content declaration
- If a child element is used in the XML document, it must be declared in a separate element type declaration
- The example (ex10.dtd) shows mixed content declaration.

```
<!ELEMENT doc-content (#PCDATA|b|i)*>
<!ELEMENT b (#PCDATA)>
<!ELEMENT i (#PCDATA)>
```

- The xml (ex10.xml) is valid according to dtd above

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE doc-content SYSTEM "ex10.dtd">
<doc-content>
    The <b>DTD</b> rules define <i>content-model</i>
</doc-content>
```

Practice 01

- You want to create XML document describe "trainees" data

→ trainees XML document will hold data more than one trainee

  → Each trainee will have

  → traineeid – id number given to each trainee

  → name – name of the trainee

  → batch - contain

    - round - round number of the course
    - course - which course is assigned to the batch

- Create a DTD document to content model for you XML document elements
- Create a sample XML document and validate it with the DTD.
- Validate the XML document
- Solution

→ DTD file (practice01.dtd)

```
<!ELEMENT trainees (trainee+)>
<!ELEMENT trainee (traineeid, name, batch)>
<!ELEMENT traineeid (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT batch (round, course)>
<!ELEMENT round (#PCDATA)>
<!ELEMENT course (#PCDATA)>
```

→ XML file (practice01.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE trainees SYSTEM "practice01.dtd">
<trainees>
 <trainee>
  <traineeid>1159699</traineeid>
<name>Mozammel</name>
  <batch>
   <round>16</round>
   <course>ESAD-CS</course>
  </batch>
 </trainee>
 <trainee>
  <traineeid>1109612</traineeid>
  <name>Mahfuz</name>
  <batch>
   <round>17</round>
   <course>ESAD-CS</course>
  </batch>
 </trainee>
</trainees>
```

Practice 02

- Consider the following facts

→ A person element have an name element

→ A person's name consists of first-name, middle-name, and last-name in order as written

→ The first-name, last-name and middle-name – each one contains text data

→ A name may not have middle-name

→ First-name, last-name can appear only once

- Write DTD code fragments for the above content model and specification.
- Write sample XML content
- Attach DTD rules as internal subset in the XML document
- Validate the xml

→ Solution

```
<?xml version="1.0"?>
<!DOCTYPE name [
 <!ELEMENT name ( first-name, middle-name?,
lastname )>
      <!ELEMENT first-name (#PCDATA) >
      <!ELEMENT middle-name (#PCDATA) >
      <!ELEMENT last-name (#PCDATA) >
      ]>
      <name>
            <first-name>Habibul</first-name>
            <last-name>Haq</last-name>
      </name>
```

Practice 03

- You have to create persons Document Type.
- Structure is described below

→ The document element is persons

→ persons contains zero one or more person elements

→ person element has an optional title element that contains text data and name element which contains text data

- Create an external DTD file
- Write an example XML file with sample data and associate the external DTD reference
- Validate the XML document
- Solution
- DTD file (practice03.dtd)

```
<!ELEMENT persons (person*)>
<!ELEMENT person ( title?, name )>
<!ELEMENT name (#PCDATA)>
<! ELEMENT title (#PCDATA)>
```

- XML file (practice03.xml)

```
<?xml version='1.0'?>
<!DOCTYPE persons SYSTEM "persons.dtd">
<persons>
    <person>
            <title>Mr.</title>
            <name>Habib</habib>
    </person>
    <person>
            <name>Shaheen Akhter</habib>
    </person> </persons>
```

Practice 04

- Consider the DTD file (practice04.dtd)
- <!ELEMENT tutorials (tutorial)+>
- <!ELEMENT tutorial (name,url)> <!ELEMENT name (#PCDATA)>
- <!ELEMENT url (#PCDATA)>
- Based on the DTD rules create a sample XML file
- Validate the XML
- Solution

```
<?xml version="1.0" ?>
<!DOCTYPE tutorials SYSTEM "practice04.dtd">
<tutorials>
  <tutorial>
    <name>XML Tutorial</name>
    <url>http://www.quackit.com/xml/tutorial</url>
  </tutorial>
  <tutorial>
    <name>HTML Tutorial</name>
    <url>http://www.quackit.com/html/tutorial</url>
  </tutorial>
</tutorials>
```

Homework 01

- You have to create a DTD rules for storing contact information in XML based on the fact below
- Document can contain many contact element
- Each contact is made of the sequence:

→ contact-name: the name of the contact
→ phone: required and can be there for many times
→ email: optional
→ address: optional

Homework 02

- You have to create a DTD rules for storing contact information in XML based on the fact below
- The document can contain 0 or more trainee element
- The trainee element contains the following element in sequence

→ ID: required
→ Name: required
→ Qualifications: made up of 1 or more degree elements

- Create a valid XML document and associate the DTD with it