DTD Basics (continued)

Chapter 03 to 04

Topics

- What are attributes?
- Working with attributes
- How to write attributes
- Attribute types
- Writing various types of Attributes
- What are Entities?
- Various types of entities
- Writing various types of attributes
- Examples
- Practice

The ATTLIST declarations identify which element types may have attributes, what type of attributes they may be, and what the default value of the attributes are .Attributes are additional information associated with an element type. They are intended for interpretation by an application.

Attributes

- Attributes are additional information associated with an element type
- They are intended for interpretation by an application

How to define attributes

- Attributes are declared using ATTLIST
- The ATTLIST declarations identify which element types may have attributes, what type of attributes they may be, and what the default value of the attributes are.
- Syntax

  <!ATTLIST element_name attribute_name attribute_type default_value | use>

→ element_name: name of the element to which the attribute applies.
→ attribute_name: name of the attribute
→ attribute_type: one of the available types supported
→ default value: default value in quotes
→ use: specifies whether the attribute is required or optional

- attribute must have either default value or use

Attribute types

- The table below list attribute types and descriptions

Attribute Type	Attribute Description
CDATA	CDATA stands for character data, that is, text that does not form markup
ID	ID is a unique identifier and its value must be unique in the document
IDREF	The IDREF value of the attribute must refer to an ID value present somewhere in the document
IDREFS	Allows multiple ID values separated by whitespace
ENTITY	A declared content
ENTITIES	Allows multiple ENTITY names separated by whitespace
NMTOKEN	The first character of an NMTOKEN value must be a letter, digit, '.', '-', '_', or ':'. It cannot contain spaces.
NMTOKENS	Allows multiple NMTOKEN names separated by whitespace
Enumerated	A list of values. The value of the attribute must be one from this list.
NOTATION	A notation name (which must be declared in the DTD).

Attribute usage

- Attribute usage signifies whether an attribute is required or not.

USE	Description
#REQUIRED	The attribute must always be included
#IMPLIED	The attribute is optional
#FIXED "value"	If the attribute is not physically added to the element tag in the XML document, the XML processor will behave as though the value does exist. If the attribute is present
	The attribute must have the value that is specified

CDATA example

- The example(ex01.dtd) shows use of CDATA

```
<!ELEMENT tutorial (title, content)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ATTLIST tutorial type #CDATA #REQUIRED>
```

- The xml (ex01.xml) is valid according to the DTD above

```
<?xml version='1.0'?>
<!DOCTYPE tutorial SYSTEM "ex01.dtd">
<tutorial type="web">
      <title>HTML5</title>
      <content>
                  Tutorial content
 </content>
</tutorial>
```

ID example

- The example (ex02.dtd) below shows use of ID type

```
<!ELEMENT tutorials (tutorial+)>
<!ELEMENT tutorial (title, content)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ATTLIST code ID #REQUIRED tutorial type #CDATA
#IMPLIED>
```

- The xml (ex02-valid.xml) is valid according to the DTD above

```
<?xml version='1.0'?>
<!DOCTYPE tutorial SYSTEM "ex02.dtd">
<tutorials>
<tutorial ID="W01" type="web">
                  <title>HTML5</title>
                  <content>
                              Tutorial content
                  </content>
</tutorial>
<tutorial ID="C01">
                  <title>LINQ</title>
                  <content>
                              Tutorial content
                  </content>
</tutorial>
</tutorials>
```

- The xml (ex02-invalid.xml) is invalid according to the DTD above as it use duplicate code attribute value

```
<?xml version='1.0'?>
<!DOCTYPE tutorial SYSTEM "ex02.dtd">
<tutorials>
<tutorial ID="W01" type="web">
                  <title>HTML5</title>
                  <content>
                              Tutorial content
                  </content>
</tutorial>
<tutorial ID="W01">
                  <title>LINQ</title>
                  <content>
```

```
                    Tutorial content
                </content>
</tutorial>
</tutorials>
```

IDREF example

- The example (ex03.dtd) below shows the use of IDREF

```
<!ELEMENT web-learning (tutorials, authors )>
<!ELEMENT tutorials (tutorial+)>
<!ELEMENT authors (author+)>
<!ELEMENT tutorial (title, content)>
<!ELEMENT author (name)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST author authorid ID #REQUIRED>
<!ATTLIST tutorial writtenby IDREF #REQUIRED>
```

- The xml (ex03-valid.xml) conforms to the DTD above

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE web-learning SYSTEM "ex03.dtd">
<web-learning>
 <tutorials>
  <tutorial writtenby="A01">
   <title>HTML5</title>
   <content>Content</content>
  </tutorial>
  <tutorial writtenby="A03">
   <title>LINQ</title>
   <content>Content</content>
  </tutorial>
 </tutorials>
 <authors>
  <author authorid="A01">
   <name>H.Hunter</name>
  </author>
  <author authorid="A02">
   <name>H.Gunter</name>
  </author>
  <author authorid="A03">
   <name>W.Butler</name>
  </author>
 </authors>
</web-learning>
```

- The xml (ex03-invalid.xml) does not conform to the DTD above as a tutorial referencing nonexistent ID

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE web-learning SYSTEM "ex03.dtd">
<web-learning>
 <tutorials>
  <tutorial writtenby="A01">
   <title>HTML5</title>
   <content>Content</content>
  </tutorial>
  <tutorial writtenby="A04">
   <title>LINQ</title>
   <content>Content</content>
  </tutorial>
 </tutorials>
 <authors>
  <author authorid="A01">
   <name>H.Hunter</name>
  </author>
  <author authorid="A02">
   <name>H.Gunter</name>
  </author>
  <author authorid="A03">
   <name>W.Butler</name>
  </author>
 </authors>
</web-learning>
```

IDREFS example

- The example(ex04.dtd) below uses IDREFS

```
<!ELEMENT web-learning (tutorials, authors )>
<!ELEMENT tutorials (tutorial+)>
<!ELEMENT authors (author+)>
<!ELEMENT tutorial (title, content)>
<!ELEMENT author (name)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST author authorid ID #REQUIRED>
<!ATTLIST tutorial writtenby IDREFS #REQUIRED>
```

- The xml (ex04.xml) is valid according to ex04.dtd

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE web-learning SYSTEM "ex04.dtd">
<web-learning>
 <tutorials>
  <tutorial writtenby="A01 A03">
   <title>HTML5</title>
   <content>Content</content>
  </tutorial>
  <tutorial writtenby="A02 A03">
   <title>LINQ</title>
   <content>Content</content>
  </tutorial>
 </tutorials>
 <authors>
  <author authorid="A01">
   <name>H.Hunter</name>
  </author>
  <author authorid="A02">
   <name>H.Gunter</name>
  </author>
  <author authorid="A03">
   <name>W.Butler</name>
  </author>
 </authors>
</web-learning>
```

NMTOKEN Example

- The example (ex05.dtd) uses NMTOKEN

```
<!ELEMENT wholesale (item, price)>
<!ELEMENT item (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

- <!ATTLIST wholesale codeno NMTOKEN #REQUIRED>
- The xml (ex04-valid.xml) is valid according to the rule above

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE wholesale SYSTEM " ex05.dtd">
<wholesale codeno="9276590-A">
 <item>Used furniture</item>
 <price>900000.00</price>
</wholesale>
```

- The xml (ex04-invalid.xml) is not valid according to the rule above as it uses invalid NMTOKEN

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE wholesale SYSTEM " ex05.dtd">
<wholesale codeno="927 6590-A">
 <item>Used furniture</item>
 <price>900000.00</price>
</wholesale>
```

Enumerated value example

- The example (ex06.dtd) below uses Enumerated values

```
<!ELEMENT message (to, subject, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<!ATTLIST message priority (High | Normal | Low)
#REQUIRED>
```

- The xml (ex06-valid.xml) is valid according the rules

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE message SYSTEM "ex06.dtd">
<message priority="High">
 <to>M. Motin, Manager (Sales)</to>
 <subject>Submiision of sales report</subject>
 <body>
      Send the last month's Sales reports by 2nd March or
before.
```

```
  </body>
</message>
```

- The xml (ex06-invalid.xml) is not valid according the rules as the priority attribute value in not in the allowable value list

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE message SYSTEM "ex06.dtd">
<message priority="Urgent">
  <to>M. Motin, Manager (Sales)</to>
  <subject>Submiision of sales report</subject>
  <body>
    Send the last month's Sales reports by 2nd March or before.
  </body>
</message>
```

Enumerated with default value example

- If default value makes the attribute optional.
- If the attribute is no present, the xml parser will add the attribute with the default value
- The example (ex07.dtd) below uses Enumerated values

```
<!ELEMENT ToDoList (task)*>
<!ELEMENT task (#PCDATA)>
<!ATTLIST task priority (important | normal) "normal">
```

- The xml (ex07.xml) is valid

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ToDoList SYSTEM "ex07.dtd">
<ToDoList>
      <task priority ="important">This is an important task.
</task>
      <task>This is by default a task that has a normal priority.
</task>
</ToDoList>
```

#REQUIRED, #IMPLIED example

- #REQUIRED makes an attribute mandatory to be included
- #IMPLIED makes an attribute optional
- The example (ex08.dtd) below shows how to use them

```
<!ELEMENT ToDoList (task)*>
<!ELEMENT task (#PCDATA)>
<!ATTLIST task priority (important | normal) #RQUIRED
status (Complete | Incomplete) #IMPLIED>
```

- The example (ex08.xml) is valid

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ToDoList SYSTEM "ex08.dtd">
<ToDoList>
      <task priority ="important">This is an important task and status not known.
</task>
      <task status="Complete">This is by default a task that has a normal priority and status Complete.
</task>
</ToDoList>
```

Entities

- An entity is a declared content that is referenced in the document.
- An entity is declared once but can be referenced many times.
- When an entity is referenced, processor retrieves the content of the entity and the content is inserted at the point where the entity is referenced
- Entities help to reduce the entry of repetitive information and also allow for easier editing.

Types of Entities

- There are two types of entity declarations: GENERAL entity declarations, and PARAMETER entity declarations
- General entities can be:

→ Internal (parsed)
→ External (parsed)
→ External (unparsed)
→ There are five entities declared in xml called pre-defined entities

Internal Entities

- Contents referenced by Internal entities reside at the location where they are declared

External Entities

- Contents referenced by external entities reside at external location not where they are declared

Parsed Entities

- Parsed generally reference text and parsed by xml processor like PCDATA

Unparsed Entities

- Unparsed entities generally reference non-XML (binary) data.
- XML processors do not parse unparsed entity contents
- External applications process unparsed contents
- Which application is required to parse a specific type of external entity is denoted by NOTATION declaration.

Declaring an internal parsed entity and referencing it

- An internal parsed entity is declared like below

```
<!ENTITY entity_name "entity content">
```

- The entity is referenced like below

```
&entity_name;
```

Internal Parsed entity example

- The example (ex09.dtd) shows how to declare internal entities

```
<!ENTITY pa "Project Authority">
<!ENTITY pc "Project Consultant">
<!ELEMENT description (#PCDATA)>
```

- The xml (ex09.xml) references the entities

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE description SYSTEM "TextFile1.dtd">
<description>
  The project is controlled by &pa; but the training aspects are maintained by &pc;
</description>
```

- The xml parser will insert the contents of the entities at the point they are referenced.
- If you open the xml in IE, you will see like below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE description …>
<description> The project is controlled by but the training aspects are maintained by </description>
```

Declaring an external parsed entity and referencing it

- An internal parsed entity is declared like below

```
<!ENTITY entity_name SYSTEM | PUBLIC "location identifier">
```

→ SYSTEM is used to specify a location in local machine and location identifier should be a file path
→ PUBLIC is used for standard location using URI which will be used the parser to find the location

- The entity is referenced like below

```
&entity_name;
```

External Parsed entity example

- Suppose you have a text file (content.txt) with the content
- The attribute must always have the default value that is specified validity constraint. If the attribute is not physically added to the element tag in the XML document, the XML processor will behave as though the default value does exist
- Now you have the DTD (ex10.dtd) like below

```
<!ENTITY c SYSTEM “content.txt”>
<!ELEMENT description (#PCDATA)>
```

- If you write the (ex10.xml) like below

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE description SYSTEM "TextFile1.dtd">
<description>
&c;
</description>
```

- If you open the xml in IE, you will see like below
  <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE description …>
  <description>The attribute must always have the default value that is specified validity constraint. If the attribute is not physically added to the element tag in the XML document, the XML processor will behave as though the default value does exist</description>

Declaring an external unparsed entity and referencing it

- External entities require extra things
- It requires a notation declaration which specifies a type and which application will process it
- Entity declaration locate the application using NDATA
- Notation is declared like below
  <!NOTATION notation_name SYSTEM | PUBLIC "location identifier">
- External entity is declared like below
- <!ENTITY entity_name SYSTEM | PUBLIC "location identifier" NDATA notation_name>

External unparsed entity example

- The example (ex11.xml) below declare external entity
  <!NOTATION gif SYSTEM "iexplore.exe">
  <!ELEMENT img EMPTY>
  <!ATTLIST img src ENTITY #REQUIRED>
  <!ENTITY logo SYSTEM "logo.gif" NDATA gif>
- The xml file (ex11.xml) uses the external entity
  <?xml version="1.0" encoding="utf-8"?>
  <!DOCTYPE img SYSTEM "ex11.dtd">

Parameter entities

- It allows you to create reusable section of text that can be used many times in DTD declarations.
- It is never referenced in XML content. It is only used within the DTD.
- It only provides certain DTD writing facilities
- Parameter entities are declared like below
  <!ENTITY % entity_name "entity content">
- Parameter entities are referenced like below
  %entity_name;

Parameter entity Example

- The example (ex12.dtd) shows how to use parameter entities
  <!ENTITY % carnv-model " (name, prey, habitat)">
  <!ENTITY % harv-model " (name, food, habitat) " >
  <!ELEMENT animals (carnivore | harbivore) >
  <!ELEMENT carnivore %carnv-model; >
  <!ELEMENT harbivore %harv-model; >
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT prey (#PCDATA)>
  <!ELEMENT food (#PCDATA) >
  <!ELEMENT habitat (#PCDATA)>
- The xml (ex12.xml) is written according to the dtd above
  <?xml version="1.0" ?>
  <!DOCTYPE animals SYSTEM "ex12.dtd">
  <animals>
  <carnivore>
  <name>Cheetah</name>
  <prey>Springbuck, Gazelle</prey>
  <habitat>African grass plain</habitat>
  </carnivore>
  </animals>

Pre-defined general entities

- XML has five text entities declared built into it.
- The table below describes them

Symbol	Predefined Entities	How to reference
<	lt	<
>	>	>
&	&	&
'	'	'
"	"	"

Practice 01

- You have to create DTD rules for trainees schema as described

Element	Content model/Attributes
Trainees	Contains one or more trainee
Trainee	traineeid - required attribute with unique value name element – contains text data course -element - contains text data batch element –contains text data
Batch	batchid element- contains textual data tsp element - contains text data

-