XML
Document Type Definition
(DTD)

1.Introduction to DTD

- *optional* DTD

  - defines XML document's grammar
  - defines structure and the legal elements and attributes of an XML document

- Purpose - to define the legal building blocks of an XML document.

- Terminology for XML:

  -- well-formed: if tags are correctly closed.

  -- valid: if it has a DTD and conforms to it.

  Validation is useful in data exchange.

Why use a DTD?

- XML provides an application independent way of sharing data. With a DTD, independent groups of people can agree to use a common DTD for interchanging data. Your application can use a standard DTD to verify that data that you receive from the outside world is valid. You can also use a DTD to verify your own data.

- A DTD can be declared inside the XML document, or as an external reference.

1) Internal DTD

This is a example of a simple XML document with an internal DTD:

<!DOCTYPE root-element [element-declarations]>

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>John</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget to meet me this weekend</body>
</note>
```

2) External DTD Declaration

If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:

<!DOCTYPE root-element SYSTEM "filename">

XML document with a reference to an external DTD

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

And here is the file "note.dtd", which contains the DTD:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Types

You can refer to an **external DTD** by either using −

- **System Identifiers** − A system identifier enables you to specify the location of an external file containing DTD declarations

contains **keyword SYSTEM** and a reference - pointing to the document's location

Syntax -

<!DOCTYPE name SYSTEM “filename.dtd" >

- **Public Identifiers** − provide a mechanism to locate DTD resources

begins with **keyword PUBLIC**

Example -

<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">

2. DTD - XML building blocks

- The building blocks of XML documents:

1)Elements: main building blocks.
Example: “company”, “ person ” …

2) Tags: are used to markup elements.

3)Attributes: Attributes provide extra information about elements. Attributes are placed inside the starting tag of an element.
Example:

4)PCDATA: parsed character data.

- character data as the *text* found between the starting tag and the ending tag of an XML element.

<name> John</name>

- PCDATA is text that will be parsed by a parser.

5)CDATA: character data.

- CDATA is *text* that will NOT be parsed by a parser.

- attribute value.

6) Entities: Entities as variables used to define common text.

- Entity references are references to entities.

The following entities are predefined in XML:

Entity References	Character
<	<
>	>
&	&
"	"
'	'

Entities are expanded when a XML document is parsed by an XML parser.

3.DTD - Elements

- In the DTD, XML elements are declared with an ELEMENT declaration

- An element declaration has the following syntax:

<!ELEMENT element-name (element-content)>

- **Element-content model**:

1) Empty Elements: keyword “EMPTY”.

<!ELEMENT element-name (EMPTY)>

example:

<!ELEMENT img (EMPTY)>

2) Text-only: Elements with text are declared:

<!ELEMENT element-name (#PCDATA)>

example:

< ! ELEMENT name (# PCDATA) >

3) Any: keyword “ANY” declares an element with any content:

<!ELEMENT element-name (ANY)>

4) Elements with Children (sequences)

- Elements with one or more children are declared with the name of the children elements inside parentheses:

<!ELEMENT element-name (child1)>

or

<!ELEMENT element-name (child1,child2,...)>

Example:

<!ELEMENT note (to,from,heading,body)>

In a full declaration, the children must also be declared, and the children can also have children. The full declaration of the "note" element is:

```
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Declaring Only One Occurrence of an Element

<!ELEMENT element-name (child-name)>

Example:

<!ELEMENT note (message)>

The example above declares that the child element "message" must occur once, and only once inside the "note" element.

Declaring Minimum One Occurrence of an Element

<!ELEMENT element-name (child-name+)>

Example:

<!ELEMENT note (message+)>

The + sign in the example above declares that the child element "message" must occur one or more times inside the "note" element.

Declaring Zero or More Occurrences of an Element

<!ELEMENT element-name (child-name*)>

Example:

<!ELEMENT note (message*)>

The * sign in the example above declares that the child element "message" can occur zero or more times inside the "note" element.

Declaring Zero or One Occurrences of an Element

<!ELEMENT element-name (child-name?)>

Example:

<!ELEMENT note (message?)>

The ? sign in the example above declares that the child element "message" can occur zero or one time inside the "note" element.

Declaring either/or Content

<!ELEMENT note (to,from,header,(message|body))>

Complex: a regular expression over other elements.

Example:

< *! ELEMENT company ((person | product)*) >

Note: The elements in the regular expression must be defined in the DTD.

This is the file “company.dtd” containing the DTD:

```
< ! 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) >
```

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

Mixed content:

Example:

<!ELEMENT note (to+,from,header,message*,#PCDATA)>

This example declares that the element **note** must contain at least one **to** child element, exactly one **from** child element, exactly one **header**, zero or more **message**, and some other **parsed character data** as well.

4.DTD - Attributes

Attributes - declared with an ATTLIST declaration

<!ATTLIST element-name attribute-name attribute-type attribute-value>

Example:

<! ELEMENT person (ssn, name, office, phone?) >

<! ATTLIST person age CDATA #REQUIRED) >

<!ATTLIST payment type CDATA "check">

XML -> <payment type="check" />

The **attribute-type** can have the following values:

Type	Description
CDATA	The value is character data
(*en1*|*en2*|..)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is a predefined xml value

The **attribute-value** can have the following values:

• value	The attribute has a default value
• #REQUIRED	The attribute is required
• #IMPLIED	The attribute is optional
• #FIXED value	The attribute value is fixed

Attribute declaration examples

Example 1: A Default Attribute Value

DTD example:

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
```

XML example:

```
<square width="100"></square>
```

or

```
<square width="100”/>
```

#REQUIRED

<!ATTLIST element-name attribute-name attribute-type #REQUIRED>

Example

DTD:

```
<!ATTLIST person number CDATA #REQUIRED>
```

Valid XML:

```
<person number="5677" />
```

Invalid XML:

```
<person />
```

#IMPLIED

<!ATTLIST element-name attribute-name attribute-type #IMPLIED>

Example

DTD:

```
<!ATTLIST contact fax CDATA #IMPLIED>
```

Valid XML:

```
<contact fax="555-667788" />
```

Valid XML:

```
<contact />
```

#FIXED

<!ATTLIST element-name attribute-name attribute-type #FIXED "value">

Example

DTD:

```
<!ATTLIST sender company CDATA #FIXED "Microsoft">
```

Valid XML:

```
<sender company="Microsoft" />
```

Invalid XML:

```
<sender company=“Google" />
```

Enumerated Attribute Values

<!ATTLIST element-name attribute-name (en1|en2|..) default-value>

Example

DTD:

```
<!ATTLIST payment type (check|cash) "cash">
```

XML example:

```
<payment type="check" />
```

or

```
<payment type="cash" />
```

5.DTD - Entities

- Entities as variables used to define shortcuts to common text
- Entity references are references to entities
- used to define shortcuts to special characters
- can be declared internal or external
- define shortcuts to special characters within the XML documents

Internal Entity Declaration

Syntax:

```
<!ENTITY entity-name "entity-value">
```

DTD Example:

```
<!ENTITY writer "Martin">
<!ENTITY copyright "Copyright XML101.">
```

XML example:

```
<author>&writer;&copyright;</author>
```

External Entity Declaration

Syntax:

<!ENTITY entity-name SYSTEM "URL">

DTD Example:

<!ENTITY writer SYSTEM "http://www.xml101.com/entities/entities.xml">

<!ENTITY copyright SYSTEM "http://www.xml101.com/entities/entities.dtd">

XML example:

<author>&writer;&copyright;</author>

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>

<!DOCTYPE address [
   <!ELEMENT address (#PCDATA)>
   <!ENTITY name “Jasmina">
   <!ENTITY company “Google">
   <!ENTITY phone_no "(011) 123-4567">
]>

<address>
   &name;
   &company;
   &phone_no;
</address>
```

Entities can be primarily of four types −

Built-in entities
Character entities
General entities
Parameter entities

Built-in entities

- All XML parsers - support built-in entities
- use these entity references anywhere
- use normal text within the XML document, such as in element contents and attribute values

- There are five built-in entities that play their role in well-formed XML, they are −
  - **ampersand: &**
  - **Single quote: '**
  - **Greater than: >**
  - **Less than: <**
  - **Double quote: "**

- Example-

```
<?xml version = "1.0"?>
<note>
  <description>I'm a technical writer & programmer</description>
<note>
```

- the & character is replaced by & whenever the processor encounters this

Character entities

- used to name - entities -> symbolic representation of information
- characters that are difficult or impossible to type can be substituted by Character Entities
- Example

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<!DOCTYPE author[
  <!ELEMENT author (#PCDATA)>
  <!ENTITY writer "Tanmay patil">
  <!ENTITY copyright "&#169;">
]>
<author>&writer;&copyright;</author>
```

- used **&#169 -** as value for copyright character
- in browser - copyright is replaced by the character ©.

General entities

- must be **declared within the DTD** before - they can be used within an XML document
- Instead of representing only a single character,
  - can represent characters, paragraphs, and even entire documents.
- **Syntax: <!ENTITY ename "text">**
- Example

```
<?xml version = "1.0"?>
<!DOCTYPE note [
  <!ENTITY writer “Martin">
]>
<note>
  &writer;
</note>
```

- XML parser - encounters a reference to ***writer*** entity,
  - supply the replacement text to the application at the point of the reference

Parameter entities

- purpose - to enable you to create reusable sections of replacement text

- **Syntax - <!ENTITY % ename "entity_value">**

- Example - Suppose you have element declarations as below

```
<!ELEMENT residence (name, street, pincode, city, phone)>
<!ELEMENT apartment (name, street, pincode, city, phone)>
<!ELEMENT office (name, street, pincode, city, phone)>
<!ELEMENT shop (name, street, pincode, city, phone)>
```

- Now suppose you want to add additional element *country*,
  - you need to add it to all four declarations
- Hence - go for a parameter entity reference

Parameter entities…

<!ENTITY % area "name, street, pincode, city">
<!ENTITY % contact "phone">

Parameter entities - dereferenced in the same way as a general entity reference, only with a percent sign instead of an ampersand –

<!ELEMENT residence (%area;, %contact;)>
<!ELEMENT apartment (%area;, %contact;)>
<!ELEMENT office (%area;, %contact;)>
<!ELEMENT shop (%area;, %contact;)>

When the parser reads these declarations, it substitutes the entity's replacement text for the entity reference.

6.DTD Validation

How to test for DTD errors while loading XML document?

Since DTD is the grammar for XML, XML is a parse tree of its DTD. Then we can use a XML parser to check if the XML is valid.

7. DTD-examples

TV Schedule DTD

```
<!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>
]>
```

Newspaper Article DTD

```
<!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">

]>
```

```
<NEWSPAPER>
<ARTICLE AUTHOR="Ravi" EDITOR="Tom" DATE="5/2/11" EDITION="vol32">
<HEADLINE>INDIA WINS THE MATCH</HEADLINE>
<BYLINE>&PUBLISHER;</BYLINE>
<LEAD>ANUJ &NEWSPAPER;</LEAD>
<BODY>INDIA BEAT DUTCH</BODY>
<NOTES>&COPYRIGHT;</NOTES>
</ARTICLE>
<ARTICLE AUTHOR="Eric" EDITOR="Robert" DATE="8/2/11" EDITION="vol39">
<HEADLINE>ARSNEL Beat MANU</HEADLINE>
<BYLINE>&PUBLISHER;  </BYLINE>
<LEAD>ANUJ &NEWSPAPER;Eric Donald</LEAD>
<BODY>INDIA BEAT DUTCH</BODY>
<NOTES>&COPYRIGHT;</NOTES>
</ARTICLE>
<ARTICLE AUTHOR="ROss" DATE="24/2/11">
<HEADLINE>Nickel back new album </HEADLINE>
<BYLINE>&PUBLISHER; </BYLINE>
<LEAD>ANUJ &NEWSPAPER;Kyra  Jhonson</LEAD>
<BODY>Someday Hit song</BODY>
<NOTES>&COPYRIGHT;</NOTES>
</ARTICLE>
</NEWSPAPER>
```

Thank You!