Document Type Definitions (DTD)

Document Type Definition (DTD)

- defines the structure and the legal elements and attributes of an XML document
- independent groups of people can agree on a standard DTD for interchanging data
- An application can use a DTD to verify that XML data is valid

An Internal DTD Declaration

```
<?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>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

An External DTD Declaration

```
<?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)>
```

External DTDs

- An external DTD (a DTD that is a separate document) is declared with a SYSTEM or a PUBLIC command:

  <!DOCTYPE RootElement SYSTEM
  "http://www.mysite.com/mydoc.dtd">
- The name that appears after DOCTYPE (here myRootElement) must match the name of the XML document's root element
- Use SYSTEM for external DTDs that you define yourself
- Use PUBLIC for official, published DTDs
- External DTDs can only be referenced with a URL
- The file extension for an external DTD is .dtd
- External DTDs are almost always preferable to inline DTDs, since they can be used by more than one document

Example-2

```
<? xml version=“1.0” ?>
<!DOCTYPE novel SYSTEM “novel.dtd”>
<novel>
    <foreword>
        <paragraph>This is the great American novel.
        </ paragraph>
    </foreword>
    <chapter number="1">
        <paragraph>It was a dark and stormy night.
        </paragraph>
        <paragraph>Suddenly, a shot rang out!
        </paragraph>
    </chapter>
</novel>
```

novel.dtd

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

The Building Blocks of XML Documents

- Elements
- Attributes
- Entities
- PCDATA
- CDATA

Elements

- Elements can contain text, other elements, or be empty.
- Examples of empty HTML elements are "hr", "br" and "img".

```
<body>some text</body>


<message>some text</message>
```

Attributes

- provide **extra information about elements**
- are always placed inside the opening tag of an element
- always come in name/value pairs
- Example:

```
<img src="computer.gif" />
```

Entities

- Some characters have a special meaning in XML, like the less than sign (<) that defines the start of an XML tag.
- An HTML entity: “ ” i.e. “no-breaking-space” entity is used in HTML to insert an extra space in a document
- Entities are expanded when a document is parsed by an XML parser.

Entities

- The following entities are predefined in XML:

Entity References	Character
&lt;	<
&gt;	>
&amp;	&
&quot;	"
&apos;	'

PCDATA

- “Parsed Character data”
- **PCDATA is text that WILL be parsed by a parser**
- **The text will be examined by the parser for entities and markup**
- Tags inside the text will be treated as markup and entities will be expanded
- Parsed character data should not contain any &, <, or > characters; these need to be represented by the & < and > entities, respectively.

CDATA

- CDATA means character data
- **CDATA is text that will NOT be parsed by a parser**
- Tags inside the text will NOT be treated as markup and entities will not be expanded.

DTD - Elements

- **Declaring Elements**
  - In a DTD, XML elements are declared with the following syntax:

```
<!ELEMENT element-name category>
or
<!ELEMENT element-name (element-content)>
```

DTD - Elements

- **Empty Elements**
  - Empty elements are declared with the category keyword EMPTY:

```
<!ELEMENT element-name EMPTY>

Example:

<!ELEMENT br EMPTY>

XML example:

<br />
```

DTD - Elements

- **Elements with Parsed Character Data**
  - Elements with only parsed character data are declared with #PCDATA inside parentheses:

```
<!ELEMENT element-name (#PCDATA)>

Example:

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

DTD - Elements

- **Elements with Any Contents**
  - Elements declared with the category keyword ANY, can contain any combination of parsable data
  - 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

DTD - Elements

- **Elements with any Contents : Example**

```
<!ELEMENT element-name ANY>

Example:

<!ELEMENT note ANY>
```

DTD - Elements

- **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)>
```

DTD - Elements

- **Elements with Children (sequences)**
  - When children are declared in a sequence separated by commas, the children must appear in the same sequence in the document
  - In a full declaration, the children must also be declared, and the children can also have children

DTD - Elements

- **Declaring Only One Occurrence of an Element**
  - child element "message" must occur once, and only once inside the "note" element.

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

Example:

<!ELEMENT note (message)>
```

DTD - Elements

- **Declaring Minimum One Occurrence of an Element**
  - child element "message" must occur one or more times inside the "note" element

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

Example:

<!ELEMENT note (message+)>
```

DTD - Elements

- **Declaring Zero or More Occurrences of an Element**
  - child element "message" can occur zero or more times inside the "note" element

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

Example:

<!ELEMENT note (message*)>
```

DTD - Elements

- **Declaring Zero or One Occurrences of an Element**
  - child element "message" can occur zero or one time inside the "note" element

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

Example:

<!ELEMENT note (message?)>
```

DTD - Elements

- **Declaring either/or Content**
  - the "note" element must contain a "to" element, a "from" element, a "header" element, and either a "message" or a "body" element

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

DTD - Elements

- **Declaring Mixed Content**
  - the "note" element can contain zero or more occurrences of parsed character data, "to", "from", "header", or "message" elements

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

DTD - Attributes

- The format of an attribute is:

```
<!ATTLIST element-name
        at_name1  type  requirement
        at_name2  type  requirement>
```

DTD - Attributes

- **Declaring Attributes**

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

DTD example:

<!ATTLIST payment type CDATA "check">

XML example:

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

Attribute Types

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

Attribute Values

Value	Explanation
value	The default value of the attribute
#REQUIRED	The attribute is required
#IMPLIED	The attribute is optional
#FIXED *value*	The attribute value is fixed

Important attribute types

CDATA — The value is character data

(man|woman|child) — The value is one from this list

ID — 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

A Default Attribute Value

- the "square" element is defined to be an empty element with a "width" attribute of type CDATA. If no width is specified, it has a default value of 0

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

Valid XML:
<square width="100" />
```

#REQUIRED

Syntax

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

Example

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

Valid XML:
<person number="5677" />

Invalid XML:
<person />
```

#IMPLIED

Syntax

```
<!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

Syntax

```
<!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="W3Schools" />
```

Entities

- There are exactly five predefined entities:

  <, >, &, ", and '
- Additional entities can be defined in the DTD:

  <!ENTITY copyright "Copyright DDU“>
- Entities can be defined in another document:

  <!ENTITY copyright SYSTEM “someURI“>
- Example of use in the XML:

  This document is &copyright; 2017.

Entities

- Entities are a way to include fixed text

- Entities should not be confused with character references, which are numerical values between `&` and `#`

- Example: `&#233;` or `&#xE9;` to indicate the character é

Another Example : XML

```
<?xml version="1.0"?>
<!DOCTYPE weatherReport SYSTEM "http://www.abc.com/doc.dtd">
<weatherReport>
        <date>20/12/2017</date>
        <location>
                <city>Nadiad</city>
                <state>GJ</state>
                <country>IN</country>
        </location>
        <temperature-range>
                <high scale="F">80</high>
                <low scale="F">64</low>
        </temperature-range>
</weatherReport>
```

doc.dtd

```
<!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>
```

Limitations of DTDs

- DTDs are a very weak specification language
  - You can't put *any* restrictions on element contents
  - It's difficult to specify:
    - All the children must occur, but may be in any order
    - This element must occur a certain number of times
  - There are only ten data types for attribute values

Limitations of DTDs

- DTDs aren't written in XML!
  - If you want to do any validation, you need one parser for the XML *and another* for the DTD
  - This makes XML parsing harder than it needs to be
  - There are newer and more powerful technologies: XML Schemas and RELAX NG
  - However, DTDs are still very much in use

References

- https://www.w3schools.com/xml/dom_intro.asp