Chapter 2

Document Type Definition (DTD)

Introduction

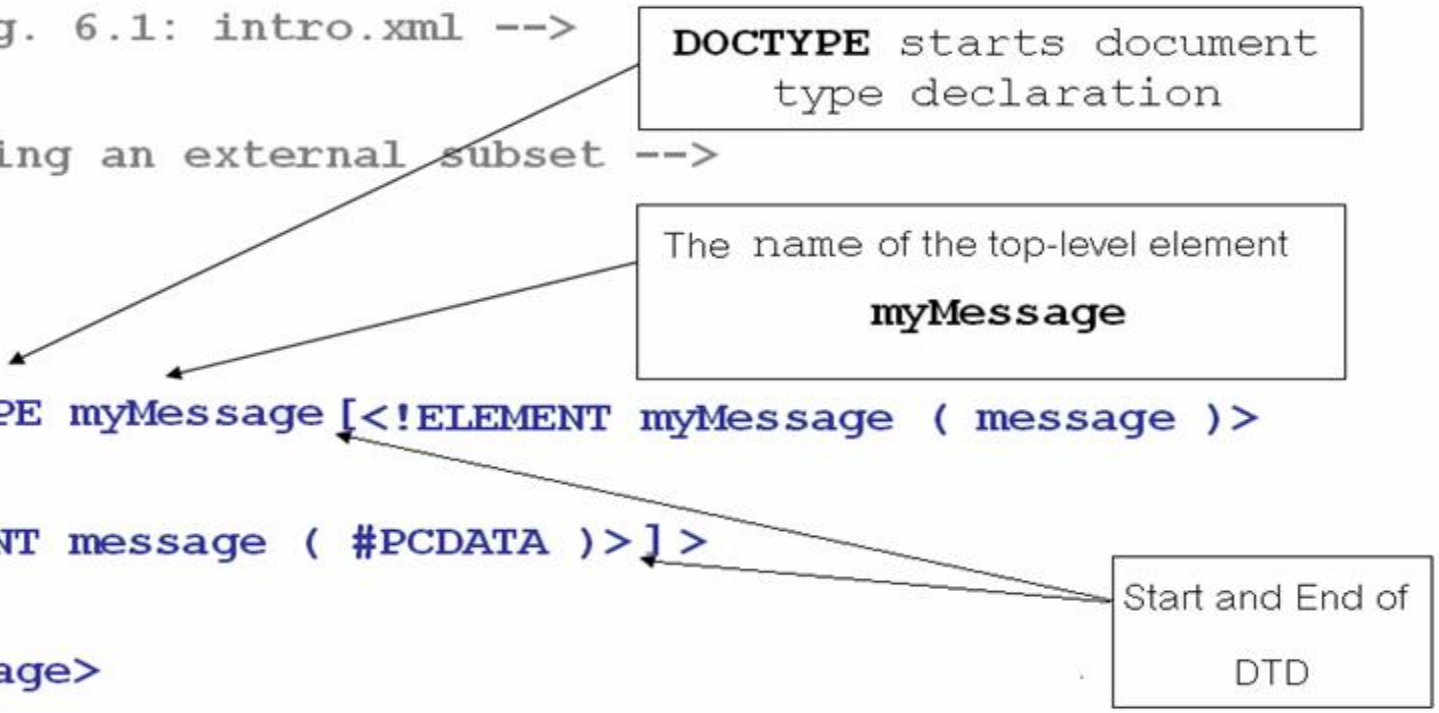
- **DTD:** Define structure of XML document.
- It defines:
  - The elements that can or must appear
  - How often the elements can appear
  - How the elements can be nested
  - Allowable, required and default attributes.

DTD Declaration

- **DTD Can be categorized as:**
  - **Internal subsets:**
    - Declarations inside document
    - Visible only within document in which it resides
  - **External subsets:**
    - Declarations outside document
    - Exist in different file
      - typically ending with `.dtd` extension

Example of Internal Subsets

```
1 <?xml version = "1.0"?>

2 <!-- Fig. 6.1: intro.xml -->

3 <!-- Using an external subset -->

4

5 <!DOCTYPE myMessage [<!ELEMENT myMessage ( message )>

6 <!ELEMENT message ( #PCDATA )>]>

7 <myMessage>

8    <message>Welcome to XML!</message>

9 </myMessage>
```



DTD Declaration cont'd

- **DTD internal subsets** :
  - We declare DTDs in XML documents using DOCTYPE Syntax.
  - This line links the XML file to the DTD subset.

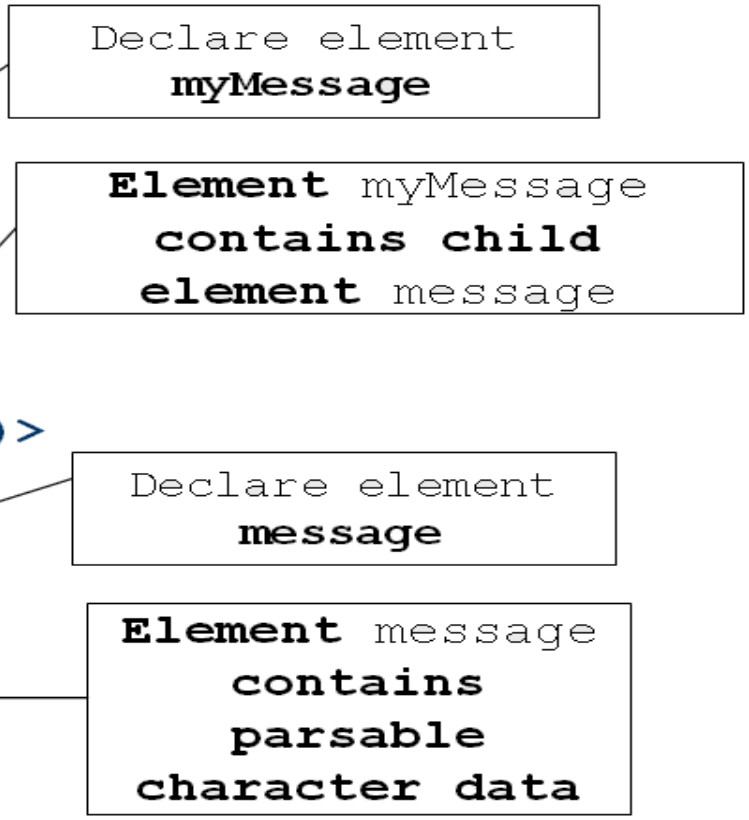
Begin with **<!DOCTYPE**

DTD –instructions ---

Ends with **>**

Example of DTD file

```
1 <!-- Fig. 6.2: intro.dtd   -->

2 <!-- External declarations -->

3 <!ELEMENT myMessage ( message )>

4 <!ELEMENT message ( #PCDATA )>
```



Example of External Subsets

```
1 <?xml version = "1.0"?>

2

3 <!-- Fig. 6.1: intro.xml -->

4 <!-- Using an external subset -->

5

6 <!DOCTYPE myMessage SYSTEM "intro.dtd">

7

8 <myMessage>

9    <message>Welcome to XML!</message>
10 </myMessage>
```

DOCTYPE starts document type declaration

The name of the top-level element

myMessage

Keyword **SYSTEM** specifies external subset

URL, Absolute path, or Relative path

Sequences

- ***Sequences:***
  - Specify order in which elements occur. (,) is a delimiter
  - **Example:**

DTD

```
<!ELEMENT Name(FName, LName)>
<!ELEMENT FName (#PCDATA)>
<!ELEMENT LName (#PCDATA)>
```

XML

```
<Name>
   <FName>Khaled</FName>
   <LName>Ghazaly</LName>
</Name>
```

Pipe Characters

- ***Choice:***
  - Specify choices. (|) is a delimiter.
  - **Example:**

DTD

```
<!ELEMENT Sport (football | baseball)>
<!ELEMENT football (#PCDATA)>
<!ELEMENT baseball (#PCDATA)>
```

XML

```
<Sport>
  <football>
      Brazil
 </football>
</Sport>
```

```
<Sport>
  <baseball>
      Brazil
 </baseball>
</Sport>
```

Occurrence Indicators

- ***Occurrence indicators:***
  - Plus sign (**+**) indicates **one** or **more** occurrences

```
<!ELEMENT Book ( chapters+ )>
```

  - Asterisk (*****) indicates **zero** or **more** occurrences

```
<!ELEMENT library ( book* )>
```

  - Question mark (**?**) indicates **zero** or **one** occurrences

```
<!ELEMENT seat ( person? )>
```

Element Type Declarations

- ***Simple declarations:***
  - Declare elements in XML documents

```
<!ELEMENT elementName (Content-Model)>
```

  - Content-Model can be:
    1. **#PCDATA**: elements contains string content only
    2. Another child **elements**
    3. **Empty**
    4. **ANY**
    5. **Mixed**

EMPTY, Mixed and ANY

3. EMPTY:

- Elements do not contain character data
- Elements do not contain child elements

DTD

```
<!ELEMENT Paragraph (MyLineBreak)>
<!ELEMENT MyLineBreak EMPTY>
```

- Markup for:

XML

```
<Paragraph>
        <MyLineBreak/>
</Paragraph>
```

EMPTY, Mixed and ANY (cont'd)

4. ANY "NOT Recommended"

- Can contain any content:
  - **#PCDATA**,
  - Elements,
  - Combination of **#PCDATA** and Elements, or
  - empty element.

```
<!ELEMENT MyCustomElement ANY>
```

- Commonly used in early DTD-development stages.

EMPTY, Mixed and ANY (cont'd)

5. Mixed "NOT Recommended"

- Combination of elements and #**PCDATA**

DTD

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

- Markup for **myMessage:**

XML

```
<myMessage> Here is some text, some
      <message> other text </message> and
      <message> even more text </message>
</myMessage>
```

Example: Mixed Element

```
1 <?xml version = "1.0" standalone = "yes"?>
2
3 <!-- Fig. 6.5 : mixed.xml          -->
4 <!-- Mixed content type elements -->
5
6 <!DOCTYPE format [
7     <!ELEMENT format ( #PCDATA | bold | italic )*>
8     <!ELEMENT bold ( #PCDATA )>
9     <!ELEMENT italic ( #PCDATA )>
10                    ]>
11
12<format>
13    This is a simple formatted sentence.
14    <bold>I have tried bold.</bold>
15    <italic>I have tried italic.</italic>
16    Now what?
17</format>
```

Specify DTD as internal subset

Declare format as mixed content element

Elements bold and italic have PCDATA only for content specification

Element format adheres to structure in DTD

Attribute Declarations

- ***Attribute declaration:***
  - Specifies element's attribute list.
  - **General form:**

```
<!ATTLIST elementName
    AttributeName AttributeTypes AttributeBehavior
    AttributeName AttributeTypes AttributeBehavior
    AttributeName AttributeTypes AttributeBehavior
    ……..
    AttributeName AttributeTypes AttributeBehavior >
```

Example

```
1 <?xml version = "1.0"?>
2
3 <!-- Fig. 6.7: intro2.xml -->
4 <!-- Declaring attributes -->
5
6 <!DOCTYPE myMessage [
7     <!ELEMENT myMessage ( message )>
8     <!ELEMENT message ( #PCDATA )>
9     <!ATTLIST message id CDATA #REQUIRED>
10>
11
12<myMessage>
13
14    <message id = "445">
15       Welcome to XML!
16    </message>
```

Specify DTD as internal subset

Declare element myMessage with child element message

Declare that attribute id contain required CDATA

Attributes Behavior

Different Attribute-Behavior :

- Mandatory represented by:
  1. #REQUIRED
- Optional represented by:
  1. Immediate "default" Values
  2. #IMPLIED
  3. #FIXED

Mandatory : REQUIRED

1. #REQUIRED:

- Attribute must appear in element.
- Document is not valid if attribute is missing.

Mandatory Example : REQUIRED

(#REQUIRED)

- *In DTD:*

```
<!DOCTYPE Document[
   <!ELEMENT Document (Customer)*>
   <!ELEMENT Customer (NAME,DATE,ORDERS)>
            ...
   <!ATTLIST Customer Credit CDATA #REQUIRED> ]>
```

- *In XML:*

```
<Document>
    <Customer Credit = "50"> ... </Customer> -→ Valid
    <Customer> ... </Customer> ----------------→ Invalid
</Document>
```

Optional: Immediate Values

1. *Immediate Values:*

- It is as a default value If the attribute's value is not present.
- It is a simple text value, enclosed in quotes.

Optional Example: Immediate Values

(Immediate "or" Default Value)

- *In DTD:*

```
<!DOCTYPE Document[
    <!ELEMENT Document (Customer)*>
    <!ELEMENT Customer (NAME,DATE,ORDERS)>
            ...
    <!ATTLIST Customer Credit CDATA "0"> ]>
```

- *In XML:*

```
<Document>
    <Customer Credit = "50"> ... </Customer> → Valid
    <Customer> ... </Customer> ----------→ Valid
</Document>
```

Optional : IMPLIED

2. #IMPLIED:

- Used when there is no default value for an attribute and you want to indicate that the author doesn't even have to use this attribute at all.
  - The programming layer decide the value of that attribute.
- This keyword used when you want to allow the author to include an attribute but not require it.

Optional Example: IMPLIED

(#IMPLIED)

- *In DTD:*

```
<!DOCTYPE Document[
    <!ELEMENT Document (CUSTOMER)*>
    <!ELEMENT Customer (NAME,DATE,ORDERS)>
        ...
    <!ATTLIST Customer Credit CDATA #IMPLIED> ]>
```

- *In XML:*

```
<Document>
  <Customer Credit ="$23.99"> ... </Customer> -→ Valid
  <Customer> ... </Customer>  ----------------→ Valid
</Document>
```

Optional : FIXED

3. #FIXED:

- They can appear in elements or not.
- It must take attribute value.
- Attribute value is constant
  - Attribute must always have that value.
  - Attribute value cannot differ in XML document.

Optional Example: FIXED

(#FIXED)

- *In DTD:*

```
<!DOCTYPE Document [
  <!ELEMENT Document (Customer)*>
  <!ELEMENT Customer (NAME,DATE,ORDERS)>
          ...
  <!ATTLIST Customer LANGUAGE CDATA #FIXED "EN"> ]>
```

- *In XML:*

```
<Document>
  <Customer> ... </Customer>  ----------------→ Valid
  <Customer LANGUAGE="EN"> . . . </Customer> → Valid
  <Customer LANGUAGE="AR"> . . . </Customer> → Invalid
</Document>
```

Attribute Types

- **Attribute Types:**

1. Strings (CDATA)
   - No constraints on attribute values
     - Except for **disallowing** **<**, **>**, **&**, **'** and **"** characters
2. ID
3. IDREF
4. NMTOKEN
5. Enumerated

Attribute Types (cont'd)

2. *ID:*

- The value is used to identify elements.
- ID value must begin with
  - a letter, underscore (_) or a colon (:)
- ID value is unique per document.
  - Mean no other ID type attribute for any element in the document can have the same value.
- You can't use the ***ID*** type with ***#FIXED*** attributes or **Immediate** values.
- Providing more than one ID attribute type **for an element** is an logical error but not DTD error.

Attribute Types (cont'd)

3. *IDREF*

- Points to elements with **ID** attribute.
- **IDREF** hold the **ID** value of another element in the document.

- **Note:**

  ID and **IDREF** must have a declared behavior of #**IMPLIED** or #**REQUIRED**.

Example

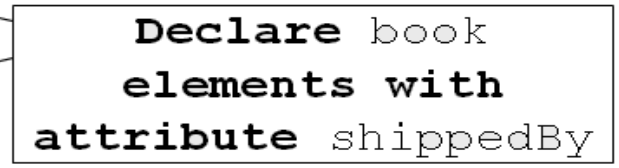
```
1 <?xml version = "1.0"?>
2
3 <!-- Fig. 6.8: IDExample.xml -->
4 <!-- Example for ID and IDREF values of attributes -->
5
6 <!DOCTYPE bookstore [
7    <!ELEMENT bookstore ( shipping+, book+ )>
8    <!ELEMENT shipping ( duration )>
9    <!ATTLIST shipping shipID ID #REQUIRED>
10   <!ELEMENT book ( #PCDATA )>
11   <!ATTLIST book shippedBy IDREF #IMPLIED>
12   <!ELEMENT duration ( #PCDATA )>
13]>
14
15<bookstore>
16   <shipping shipID = "s1">
17      <duration>2 to 4 days</duration>
18   </shipping>
19
```

Each shipping element has a unique identifier (shipID)

Attribute shippedBy points to shipping element by matching shipID attribute1

Example (cont'd)

```
20    <shipping shipID = "s2">
21       <duration>1 day</duration>
22    </shipping>
23
24    <book shippedBy = "s2">
25       Java How to Program 3rd edition.
26    </book>
27
28    <book shippedBy = "s2">
29       C How to Program 3rd edition.
30    </book>
31
32    <book shippedBy = "s1">
33       C++ How to Program 3rd edition.
34    </book>
35</bookstore>
```



Attribute Types (cont'd)

4. NMTOKEN:

- Tokenized attribute type
- "Name token"
- Value consists of letters, digits, periods, underscores, hyphens and colon characters
- It Just Take One word,
  - it can't contain spaces, comma.

Example

IN DTD:

```
<!ELEMENT company (#PCDATA)>
<!ATTLIST company Address NMTOKEN #REQUIRED>
```

In XML:

```
<company Address = "241 ElAhram St">
```
not valid

```
<company Address = "241_ElAhram:St">
```
valid

Attribute Types (cont'd)

5. Enumerated:

- Declare list of possible values for attribute

```
<!ATTLIST person gender ( M | F ) "F">
```

  - Attribute **gender** can have either value **M** or **F**
  - **F** is default value.

ENTITY

- Entity is a place to store text data, like constant in JAVA.

- **General Entity declaration:**

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

ENTITY

- **Example:**

In DTD:

```
<!ENTITY My_Address "241 El-Haram st. Giza">
```

In XML:

```
<address>&My_Address;</address>
```

– Entity reference `&My_Address;` replaced by its value

```
<address>241 El-Haram st. Giza</address>
```

ENTITY

- **External General Entity declaration:**

```
<!ENTITY My_Address SYSTEM "My_Addr.ent">
```

  - **In file My_Addr.ent:**
    - `241 El-Haram st. – Giza`
  - Entity may be used as follows:

```
<useAnEntity>&My_Address;</useAnEntity>
```

  - Entity reference **&My_Address;** replaced by its value

```
<useAnEntity> 241 El-Haram st. – Giza</useAnEntity>
```

Assignment

- Design a DTD of the configuration file for a library that you made.

- Note:
  - External DTD
  - In DTD:
    - Define elements with occurrence indicators
    - Define attribute with different types (CDATA, enumerated,…) with different behavior (required, optional