4.5 DOCUMENT TYPE DECLARATION (DTD) and XML SCHEMAS

The document type declaration attaches a DTD to a document. It is a way to describe XML language precisely.

<!DOCTYPE element DTD identifier

[declaration1

declaration2

.....]>

The DTD starts with <!DOCTYPE delimiter. An element tells the parser to parse the document from the specified root element. DTD identifier is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called **External Subset.** The square brackets [ ] enclose an optional list of entity declarations called **InternalSubset.**

Internal DTD

A DTD is referred to as an internal DTD if elements are declared within the XML files. To refer it as internal DTD, standalone attribute in XML declaration must be set to yes.

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

root-element is the name of root element and element-declarations is where the user declare the elements.

Internal DTD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE address [
<!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>]>
<address> <name>Sharanya</name>
<company>abc</company>
<phone>12347890</phone> </address>
```

Start Declaration- Begin the XML declaration with following statement

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

DTD- Immediately after the XML header, the document type declaration follows, commonly referred to as the DOCTYPE:

<!DOCTYPE address [

The DOCTYPE declaration has an exclamation mark (!) at the start of the element name. The DOCTYPE informs the parser that a DTD is associated with this XML document.

DTD Body- The DOCTYPE declaration is followed by body of the DTD, where elements, attributes, entities, and notations are declared.

End Declaration - Finally, the declaration section of the DTD is closed using a closing bracket and a closing angle bracket (]>).

Rules

- The document type declaration must appear at the start of the document.
- The element declarations must start with an exclamation mark.
- The Name in the document type declaration must match the element type of the root element.

External DTD

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

<!DOCTYPE root-element SYSTEM "file-name">

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<!DOCTYPE address SYSTEM "address.dtd">

<address> <name>Sharanya</name>

<company>abc</company>

<phone>1234689</phone> </address>

address.dtd

<!ELEMENT address (name,company,phone)> <!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)> <!ELEMENT phone (#PCDATA)>

Types of external DTD

- **System Identifiers:** A system identifier enables the user to specify the location of an external file containing DTD declarations.

```
<!DOCTYPE name SYSTEM "address.dtd" [...]>
```

- **Public Identifiers:** Public identifiers provide a mechanism to locate DTD resources.

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

Public identifiers are used to identify an entry in a catalog. Public identifiers can follow any format, however, a commonly used format is called **Formal Public Identifiers, or FPIs**.

Advantages of DTD

- The XML processor enforces the structure, as defined in the DTD.
- The application easily accesses the document structure.
- The DTD gives hints to the XML processor—that is, it helps separate indenting from content.
- The DTD can declare default or fixed values for attributes. This might result in a smaller document.

XML SCHEMAS

XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="contact">
<xs:complexType><xs:sequence>
<xs:element name="name" type="xs:string" />
<xs:element name="company" type="xs:string" />
<xs:element name="phone" type="xs:int" /> </xs:sequence>
</xs:complexType></xs:element> </xs:schema>
```

- **Elements:** An element can be defined within an XSD as follows:

```
<xs:element name="x" type="y"/>
```

- **Definition Types**

  - **Simple Type** - Simple type element is used only in the context of the text. Some of predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date.

    Example: <xs:element name="phone_number" type="xs:int" />

  - **Complex Type** - A complex type is a container for other element definitions. This allows the user to specify which child elements an element can contain and to provide some structure within the user's XML documents.

```
<xs:element  name="Address">
<xs:complexType><xs:sequence>
<xs:element  name="name"  type="xs:string"  />
<xs:element name="company" type="xs:string" />
<xs:element  name="phone"  type="xs:int"  />
</xs:sequence><xs:complexType>
</xs:element>
```

  - **Global Types** - With global type, the user can define a single type in the user's document, which can be used by all other references.

```
<xs:element  name="AddressType">
<xs:complexType><xs:sequence>
<xs:element  name="name"  type="xs:string"  />
<xs:element name="company" type="xs:string" />
</xs:sequence></xs:complexType>
</xs:element>
```

Now let us use this type in our example as below:

```
<xs:element  name="Address1">
<xs:complexType><xs:sequence>
<xs:element  name="address"  type="AddressType"  />
<xs:element  name="phone1" type="xs:int" />
</xs:sequence></xs:complexType>
```

```
</xs:element>
<xs:element name="Address2">
<xs:complexType><xs:sequence>
<xs:element name="address" type="AddressType" />
   <xs:element name="phone2" type="xs:int" />
</xs:sequence></xs:complexType>
</xs:element>
```

Instead of having to define the name and the company twice (once for Address1 and once for Address2), we now have a single definition.

- **Attributes**

Attributes in XSD provide extra information within an element. Attributes have name and type property as shown below:

```
<xs:attribute name="x" type="y"/>
```