Web Technology

Unit-3
XML

Outline

1. Introduction
   - Introduction to XML
   - Features of XML
   - XML Key Component
2. Document Type Definition (DTD)
3. XML Schemas
4. XSL
5. XSLT

Introduction to XML

- XML stands for e**X**tensible **M**arkup **L**anguage
- XML is a language to describe other languages.
- Its main purpose is to allow the sharing of data across different type of systems and it is particularly useful in this sense for applications that do this over the internet.
- Example :

```
<?xml version="1.0">
<person>
        <first>Narendra</first>
        <last>Modi</last>
        <birthdate>01/01/45</birthdate>
        <employed started="01/02/03">
                Prof. @ Darshan college
        </employed>
</person>
```

Here person is root element

Here started is an attribute of element employed

Features of XML

- It is in a format that both **human** and **machines** can read.
- It supports **Unicode**.
- It supports **data structures**.
- It is **self-documenting**.
- It has a **strict format** that makes it easy for **parsing** to take place.
- It can be understood and exchanged between **dissimilar systems**.
- It can be useful for swapping data between **different applications**.

XML Key Component

- One of the key aspects of XML is how strict the syntax is.
- There are mainly 3 components of the XML
  1. Elements
  2. Attribute
  3. Namespace

1) Elements

- The strict syntax of XML contains a few rules about elements that must be adhered to:
  - Elements must have a **closing tag**.
  - Tags are **case sensitive**
  - Elements must be **nested correctly**
  - XML documents must have a **root element**.
- Example :

<**birthdate**>26th October 1788</**birthdate**> ✓

<Birthdate>26th October 1788</birthdate> ✗ (elements are case sensitive)

Hello ✗ (elements not nested properly)

Hello ✓

2) Attributes

- Attributes can be added to elements in XML but must always be quoted.
- For Example, here employed is a element and started is the attribute of the element employed.

<employed started="10/11/12">Darshan, Rajkot </employed> ✓

<employed started=10/11/12>Darshan, Rajkot </employed> ✗

Value must be quoted

3) Namespace

- Sometimes in XML there is a danger of conflicting names between documents.
- Example :
  - You create one document with name element for the professor, it may also possible someone else create a document with name element for the animal name, so to avoid the conflict we can use namespaces.
- Namespace usually take the form of a URL, beginning with a domain name, an optional namespace label in the form of a directory name and finally a version number, which is also optional.

  xmlns = “http://www.mydomain.com/ns/animals/1.1”

3) Namespace (Example)

- This XML carries HTML table information:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

- This XML carries information about a table (a piece of furniture):

```
<table>
  <name>Saag table</name>
  <width>3</width>
  <length>6</length>
  <weight>5kg</weight>
</table>
```

3) Namespace (Example) (Cont.)

- To solve the conflict problem we can use namespace of html table.
- Example :

```
<h:table
xmlns:h="http://www.w3.org
/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

- To solve the conflict problem we can use namespace of furniture table.
- Example :

```
<f:table
xmlns:f="http://darshan.ac.in/fu
rntiture">
  <f:name>Saag table</f:name>
  <f:width>3</f:width>
  <f:length>6</f:length>
  <f:weight>5kg</f:weight>
</f:table>
```

XML Key Component (Cont.)

- **White space** is **preserved** in XML where as in HTML it is truncated down to just a single space.
- One thing does remain in common with HTML though is comments,
  - Comments can be added using the triangle brackets like this:

    <!-- here are some remarks -->

Document Type Definition (DTD)

- XML is particularly concerned with being well formed or correct in syntax.
- There are two ways of checking whether the document follows expected order and structure
  - **Document Type Definitions (DTDs)**
  - **Schemas**
- A Document Type Definition (**DTD**) defines the legal building blocks of an XML document.
- A DTD can be declared **inline** inside an XML document, or as an **external** reference

Why Use a DTD?

- With a **DTD**, each of your XML files can carry a **description** of its **own format.**
- With a DTD, independent groups of people can agree to use a **standard DTD for interchanging data.**
- Your application can use a standard DTD to **verify** that the data you **receive** from the outside world is **valid.**

DTD (Example)

```
<?xml version="1.0"?>
<!DOCTYPE note [
    <!ELEMENT note (to,from,title,message)>
    <!ELEMENT to (#PCDATA)>
    <!ELEMENT from (#PCDATA)>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT message (#PCDATA)>
]>
```

DTD (Cont.)

- DTD can be internal or external
- If it is **internal** than simply put previous code to the **top of the XML** file
- If it is external than save it as **.dtd** file extension and refer it from XML,

<!DOCTYPE note SYSTEM “note.dtd”>

DTD Elements

- We can specify the number of occurrences of the elements using +, *, ? and | operators (works ~ similar to Regular Expression)
- Example :
  - <!ELEMENT note(to+,from,title?,message*) />
- Above example suggest that **root** element of the xml must be **note** and should have one or more (+) **recipients**, **sender** should be only one, **title** must be one or zero(?) and **messages** can be zero or more(*).
- We can also specify to have either one of the elements using | operator
  - <!ELEMENT note(to,from,title,message|information) >
- In above declaration we have specified that either message should be there or the information element should be there in the note

DTD Attribute

We can specify the attributes also using DTD using **ATTLIST** declaration.

- Syntax:

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

- Example :

  <!ATTLIST employed started CDATA “01/01/01”>

This is the default value for started

We can also specify required or fixed for the attribute

- Example :

  <!ATTLIST employed started CDATA #REQUIRED>

  <!ATTLIST employed started CDATA #FIXED “01/01/01”>

This suggest that started is mandatory field

XML Schema

- XML **Schema** is an XML-based **alternative** to **DTD**
- An XML schema describes the structure of an XML document.
- The XML Schema language is also referred to as **XML Schema Definition** (**XSD**)
- An XML Schema
  - defines **elements** that can appear in a document
  - defines **attributes** that can appear in a document
  - defines which elements are **child elements**
  - defines the **order** of child elements
  - defines **data types** for elements and attributes
  - defines **default** and **fixed** values for elements and attributes

XML Schema (cont.)

- XML Schemas are the Successors of DTDs
  - XML Schemas are **extensible** to future additions
  - XML Schemas are **richer** and **more powerful** than DTDs
  - XML Schemas are **written** in **XML**
  - XML Schemas support **data types**
  - XML Schemas support **namespaces**

XML Schema (Example)

note.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema >
    <xs:element name="note“>
        <xs:complexType>
          <xs:sequence>
              <xs:element name="to" type="xs:string"/>
              <xs:element name="from" type="xs:string"/>
              <xs:element name="heading" type="xs:string"/>
               <xs:element name="body" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XML Schema (Example) (cont)

```
<?xml version="1.0"?>
<note
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="note.xsd">
    <to>Darshan</to>
    <from>Student</from>
    <heading>Reminder</heading>
    <body>Don't forget to attend lecture thisweekend!</body>
</note>
```

Data Types in XSD

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time
- Etc…..

Complex Types in XSD

- Complex elements can be built that contain other elements and attributes.
- For example,

```
<xs:complexType name="productinfo">
    <xs:sequence>
      <xs:element name="item" type="xs:string" />
      <xs:element name="itemcode" type="xs:string" />
    </xs:sequence>
</xs:complexType>
<xs:element name="food" type="productinfo"/>
<xs:element name="magazine" type="productinfo"/>
<xs:element name="clothes" type="productinfo"/>
```

Default ,Fixed and Required Values

- To define attribute a similar style is adopted, for example for the XML element :

  <firstname lang="English">Narendra</firstname>

- XSD would be

  <xs:attribute firstname="lang" type="xs:string"/>

- Default value attribute in XSD

  <xs:attribute firstname="lang" type="xs:string" **default="EN"**/>

- Fixed value attributes in XSD

  <xs:attribute firstname ="lang" type="xs:string" **fixed="EN"**/>

- Required value attributes in XSD

  <xs:attribute firstname ="lang" type="xs:string" **use="required"**/>