BCA (2014-2017) Client Side Web Technology XML Notes

Xml Introduction

- XML that stands for eXtensible Markup Language is a meta-markup language.
- XML was designed to describe, transport and store data.
- XML tags are not predefined. You must define your own tags
- The Extensible Markup Language (XML) is a document processing standard that

  is an official recommendation of the World Wide Web Consortium (W3C).
- XML is not a replacement for html

XML Features

- If your data is “Anuja Kale Pune”, you have no context to it.
- In html it would be :

```
<h1>
 Anuja <br/> Kale<br/> <br/> Pune <br/>
</h1>
```

- xml is self describing

```
 <person>
     <name>Anuja</name>
     <surname>Kale/surname>
     <city>Pune</city>
</person>
```

Differences between HTML and XML

- HTML was designed to *display data* and to focus on *how data looks*.
- You must use predefined tags
- Html is case insensitive
- When HTML is used to display data, the data is stored inside your HTML.
- It is content unaware
- Does not check syntax

- XML was designed to *describe data* and to focus on *what the data is*
- You have to define your own tags in xml
- XML is case sensitive
- XML can Separate Data from HTML. With XML, your data is stored outside your HTML
- Xml is content aware
- Does a syntax check

Elements in XML

- An XML element is everything from (including) the element's start tag to (including) the element's end tag.

  Ex : ***<mytag>hello</mytag>***
- Element can contain text or other xml elements
- XML Elements are extensible and they have relationships.
- XML Elements have simple naming rules.
  - Names can contain letters, numbers, and other characters
  - Names cannot start with a number or punctuation character
  - Names cannot start with the letters xml (or XML, or Xml, etc)
  - Names cannot contain spaces
  - Any name can be used, no words are reserved
- XML Elements can also have attributes.
- XML documents can be extended to carry more information.

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- PRODUCT DATTA IS THE ROOT ELEMENT-->

<productdata>
    <product prodid="p001">
            <productname> Barbie Doll </productname>
            <description>This is the toy for children.</description>
            <price>Rs.20.00</price>
            <quantity>20</quantity>
    </product>

<product prodid="p002">
            <productname> Monkey </productname>
            <description>This is monkey for children.</description>
            <price>Rs.30.00</price>
            <quantity>30</quantity>
    </product>
</productdata>
```

Processing Instruction

- A type of tag supported by XML
- Declares information necessary for processing a document, or directs any program that processes the document to perform a specific function
- Starts with <? and ends with ?>
- Example
  - <?xml version=“1.0”?>
- Provides information about a specific application

- The <?xml version=“1.0” ?> declaration is necessary for an software to identify an xml document.
- The other attributes of this processing instruction are :
  - encoding : specifies the character sets used like ISO-8859-1(latin-1 west european), UTF-8 etc. Default is UTF-8.
  - standalone : specifies if this xml doc also refers to other external docs like dtd or css/xsl etc or is self sufficient. If standalone is yes, it cannot refer to any other dtd or stylesheet file.
  - Version is an mandatory attribute

Well- formed XML

- Any xml that follows all the below rules is said to be a well formed XML.
  1. Must contain at least one element. Must have a root element.
  2. Every start tag must have a corresponding end tag.
  3. All tags must be properly nested.
  4. Tags in xml are case sensitive.
  5. Attribute values must be quoted.
  6. Element names can begin with a char or underscore _. Subsequent characters in the name may include letters, digits, underscores, hyphens, and periods.

Create an xml for the following data given

- Company
  - Name
  - Location
  - Departments
    - Department
      - DName
      - Manager
        - » DateOfJoining
        - » Salary

There are more than one departments in the company.

```xml
<Company>
    <Name>ABC corp</Name>
    <Location>Pune</Location>
    <Departments>
        <Department>
                <Dname>Finance</Dname>
                <Manager>
                        <DateOfJoining>23/03/1998</DateOfJoining>
                        <Salary>500000</Salary>
                </Manager>
        </Department>
        <Department>
                <Dname>HR</Dname>
                <Manager>
                        <DateOfJoining>29/08/1999</DateOfJoining>
                        <Salary>300000</Salary>
                </Manager>
        </Department>
    </Departments>
</Company>
```

XML-DTD

Document Type Definition - DTD

- The purpose of DTD is to define the legal building blocks of any XML document
- It defines the structure of a xml document . It defines the document structure with a list of legal elements and attributes.
- defines what each XML element will contain, i.e. text, child element and so on and also defines the order and nesting of the XML elements.
- DTD usage :
  - defines a structure for the xml document
  - each xml can have the description of its own format
  - different groups of people can interchange data by agreeing on a particular XML format which can be validated using a DTD.
  - Any application can verify the data it receives from outside against a pre defined dtd.

XML Parsers

- A *parser* is a piece of program that takes a physical representation of some data and converts it into an in-memory form for the program as a whole to use.
- One of the ways to classify XML parsers is as below :
  - non-validating**:** the parser does not check a document against any DTD (Document Type Definition); only checks that the document is *well-formed* (that it is properly marked up according to XML syntax rules)
  - validating**:** in addition to checking well-formedness, the parser verifies that the XML document conforms to a specific DTD i.e. the xml file is written as per defined by the DTD.

- An xml document is well formed if it follows all the rules for well formedness.
- It is valid if it is well formed and follows a DTD.
- So,
- *All valid xml documents are well formed but all well formed documents may not be valid.*

Xml building blocks

- Any xml file can contain the following types of elements:
  - an empty element :

    ```
    <letter></letter>
    ```
  - Element with attribute :

    ```
    <letter Date="2009/11/11"></letter>
    ```
  - Element with text :

    ```
    <letter>This is the letter content</letter>
    ```
  - Element with child elements :

    ```
    <letter>
            <sender>Nilofer</sender>
     </letter>
    ```
- An element in an DTD file is represented as :
  - **<!ELEMENT ELEMENT_NAME CONTENT_MODEL>**
    - Name : is the name of the element you are defining.
    - Content model : specifies what the xml element can contain. Following are the different types of content models

CONTENT_MODEL : ANY

- The first type of content model is :
  - Any : indicates that the element can have any type of content.
    - which means the element content will not be validated by the xml parser.
    - **<!ELEMENT PRODUCT ANY>**
    - Removes the syntax checking associated with the element.
    - If you put any child elements in the element of ANY content type, those child elements must be declared in the dtd.

CONTENT_MODEL : CHILD ELEMENTS

- Child elements : you can specify any element to contain other XML elements.
- Ex: The XML element BOOK is as below :

```
<BOOK>
   <CHAPTER>introduction</CHAPTER>
 </BOOK>
```

- The DTD declaration for a BOOK element is

  <!ELEMENT BOOK (CHAPTER)>

- If BOOK contains more than one child elements

  Ex :
  ```
  <BOOK>
              <CHAPTER>one</CHAPTER>
              <PRICE>20.99</PRICE>
      </BOOK>
  ```
  ```
  <!ELEMENT BOOK (CHAPTER,PRICE)>
  ```
- The order of the chapter and price must be followed as written in the DTD definition.
- Illegal Definition
  ```
  <BOOK>
      <PRICE>20.99</PRICE>
      <CHAPTER>one</CHAPTER>
  </BOOK>
  ```

CONTENT_MODEL : CHILD ELEMENTS Contd.

- For defining multiple occurrences of child elements
  - (CHAPTER)+ : indicates one or more occurrences of chapter element. **<!ELEMENT BOOK (CHAPTER)+>**
  - (CHAPTER)? : zero or one occurrences of chapter.

    <!ELEMENT BOOK (CHAPTER)?>
  - CHAPTER | APPENDIX : either of the two can occur.

    <!ELEMENT BOOK (CHAPTER|APPENDIX)>

CONTENT_MODEL : PCDATA

- An element can contain text, for which the content model used would be :
- #PCDATA : is the non markup text.
- The parsed character data(PCDATA) is the actual content of the xml document, the plain text.
- PCDATA will allow an element to contain only text but NOT other child elements.
- PCDATA is not able to distinguish between the types of data like numbers, characters, floats etc

Content Model : PCDATA

- <!ELEMENT BOOK(CHAPTER)>
  <!ELEMENT CHAPTER(#PCDATA)>
- For the above dtd declaration a valid xml would be

```
<BOOK>
   <CHAPTER>
    Introduction to xml dtds
   </CHAPTER>
</BOOK>
```

Some dtd and xml examples

DTD

- <!ELEMENT BOOK (CHAPTER+,APPPENDIX?) >

  <!ELEMENT CHAPTER (#PCDATA) >

  <!ELEMENT APPPENDIX(#PCDATA) >

XML

- <BOOK>

  <CHAPTER>1</CHAPTER>

  <APPENDIX> E</APPENDIX>

  </BOOK>
- **OR**
- <BOOK>

  <CHAPTER>1</CHAPTER>

  <CHAPTER>2</CHAPTER>

  <APPENDIX> E</APPENDIX>

  </BOOK>
- **OR**
- <BOOK>

  <CHAPTER>1</CHAPTER>

  <CHAPTER>2</CHAPTER>

  </BOOK>

DTD

- Subsequences using parenthesis :

  <!ELEMENT CHAPTER (NAME, (PAGENO,REFERENCE*)+) >

XML

-
  ```
  <CHAPTER>
  <NAME>JIM</NAME>
  <PAGENO> 23</PAGENO>
  <REFERENCE>a </REFERENCE>
  </CHAPTER>
  ```
- OR

-
  ```
  <CHAPTER>
  <NAME>JIM</NAME>
  <PAGENO> 23</PAGENO>
  <REFERENCE>a </REFERENCE>
  <PAGENO> 29 </PAGENO>
  <REFERENCE>C </REFERENCE>
  </CHAPTER>
  ```
- OR

-
  ```
  <CHAPTER>
  <NAME>JIM</NAME>
  <PAGENO> 23</PAGENO>
  <PAGENO> 90</PAGENO>
  </CHAPTER>
  ```

DTD

- Choice :

  <!ELEMENT CHAPTER (NAME, PAGENO, (REFERENCE|FOOTNOTE))

XML

- <CHAPTER>

  <NAME>JIM</NAME>

  <PAGENO> 23</PAGENO>

  <REFERENCE>a </REFERENCE>

  </CHAPTER>
- OR
- <CHAPTER>

  <NAME>JIM</NAME>

  <PAGENO> 23</PAGENO>

  <FOOTNOTE>FT</FOOTNOTE>

  </CHAPTER>

CONTENT_MODEL : MIXED

- An element can contain both child elements and text, in which case the content model would be mixed.
- It does not allow the user to specify the order or number of occurrences of child elements.
- <!ELEMENT chapter (#PCDATA | NoofPages)*>
- The corresponding Xml will be

```
<chapter>                                     <chapter>
<NoofPages> 20 </NoofPages>   or             <NoofPages> 20 </NoofPages>
introduction to xml                          </chapter>
     </chapter>
```

- The element can contain either or both of the mixed content for any number of times.

CONTENT_MODEL : EMPTY

- Empty elements do not have any text or child element BUT they may have attributes;
- Ex : <Book_id></Book_id>
- In the dtd,
- <!ELEMENT Book_id EMPTY>

DTD Types

- Internal DTD :
  - Xml and DTD are in the same file
  - <! DOCTYPE root-element [element-declarations] >
- External DTD :
  - A dtd that resides in a file (.dtd extension) other than the xml file
  - Advantage of an external dtd is reusability
  - Can be
    - Private : shared by a group of people.
      - <!DOCTYPE root-element SYSTEM "filename“>
      - In place of filename you can specify an url also
    - Public : when the dtd is intended for public use
      - <!DOCTYPE root-element PUBLIC “dtd_name” “dtd_URL”>

Internal dtd

- **Sample xml file with an internal dtd :**

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

<!DOCTYPE book [
<!ELEMENT book (title,noofpages,price,edition)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT noofpages (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
]>

<book>
<title>Xml Pocket reference</title>
<noofpages>150</noofpages>
<price>100.00</price>
<edition>second</edition>
</book>
```

External dtd

- Sample xml file with external dtd

```
<?xml version="1.0"?>
<!DOCTYPE book SYSTEM “test.dtd”>
<book>
<title>Xml Pocket reference</title>
<noofpages>150</noofpages>
<price>100.00</price>
<edition>second</edition>
</book>
```

Having internal and external dtds

- You can declare some elements in internal dtd and remaining in external dtd.
- But declaring same element both in internal and external dtds may not be acceptable by some processors while some may allow internal dtds to take precedence.

DTD Attributes

- Attributes provide supplementary information about a particular element and appear in name/value pairs.
- <book id="123"></book>
- In dtd, you define an attribute as below :
  - <!ATTLIST ELEMENT_NAME
    ```
    ATTRIBUTE_NAME   TYPE   DEFAULT_VALUE
    ATTRIBUTE_NAME   TYPE   DEFAULT_VALUE
    >
    ```
- <!ATTLIST book id CDATA "123">
- where ELEMENT_NAME and ATTRIBUTE_NAME is the name of the element and its attributes resp.

Attribute Default values

- You can specify the actual default value in DTD:
  - <!ATTLIST PAGE AUTHOR CDATA “AUTHOR_ONE”>
- The corresponding xml element would be
  - <PAGE AUTHOR=“AUTHOR_ONE” />

  Or

  - <PAGE AUTHOR=“Dan Brown” />

Attribute Types

- Type specifies the type of the attribute.
- Some important attribute types are :
  - CDATA : plain text/character data not including any markup.
    - <!ATTLIST book id CDATA “000”>
  - Enumerated : a list of possible values out of which any one would be the attribute value.
    - <!ATTLIST emp employment_status (contract | regular) “regular”>