XML

eXtensible Markup Language

Japanese

How can we communicate ?

French

Application A
Using Java
Oracle DB
Linux OS

XML

Application B
Using .NET
DB2
Windows OS

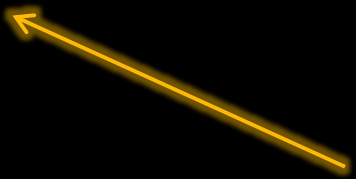
INTRODUCTION

- XML is a Markup Language
- Derived from SGML
- Designed to Store and Transport Data
- "Defines a set of rules for encoding documents in a format that is both Human Readable and Machine Understandable"
- Used to exchange data or information across the Internet.
- Used to describe the Structure of a Document not the way it is presented.

- Data is stored in Plain Text format
- Saved as .xml
- Can define your own Tags : User-defined Tags
- Case Sensitive Language.
- Designed to be Portable and Platform Independent.

STRUCTURE OF AN XML DOCUMENT

```
<?xml version="1.0" encoding="UTF-8" ?>
<rootelement>
    <childelement>
        <subchild>  -  </subchild>
        .
        .
    </childelement>
</rootelement>
```



XML Declaration
(Mandatory)

- XML Document mainly consists of 2 parts:
  - **XML Prolog**
  - **XML Body – Consists of elements and attributes**
- The ***prolog*** of an XML document comprises **everything from the start of the file to the document root tag**. It may contain the
  - **XML declaration,**
  - **Processing Instructions,**
  - **Comments, and a**
  - **DTD.**
- The first line in an XML file is the **XML declaration**.
  - identifies the document as being XML.
  - defines the **XML standard version** the label adheres to, and also
  - defines the **character set** to be used.

XML Syntax rules:

- XML documents must contain **one root element** that is the **parent** of all other elements.
- The **XML prolog** is optional. If it exists, it must come first in the document.
- All XML **Elements Must Have a Closing Tag**
- XML **Tags are Case Sensitive**
- XML **Elements Must be Properly Nested** - Closed in the reverse order in which they are opened
- XML **Attribute Values Must Always be Quoted**
- **White-space is Preserved in XML** and it does not truncate multiple white-spaces
- XML documents that conform to the syntax rules above are said to be **"Well Formed"** XML documents.

Example XML Document

- Catalogue.xml

```
<?xml  version="1.0" ?>
<catalogue>
    <book  id="1">
        <title>XML Bible</title>
        <author> Watson </author>
        <isbn>123-456-789</isbn>
        <publication>TMH</publication>
        <edition>3</edition>
        <price>50 $</price>
    </book>
</catalogue>
```

XML Declaration

Root Element

Attribute

Start tag

End tag

C:\Users\ONLINE\Desktop\Book.xml

C:\Users\ONLINE\Desktop\...

```
<?xml version="1.0"?>
- <catalogue>
  - <book id="1">
      <title>XML Bible</title>
      <author> Watson </author>
      <isbn>123-456-789</isbn>
      <publication>TMH</publication>
      <edition>3</edition>
      <price>50 $</price>
    </book>
</catalogue>
```

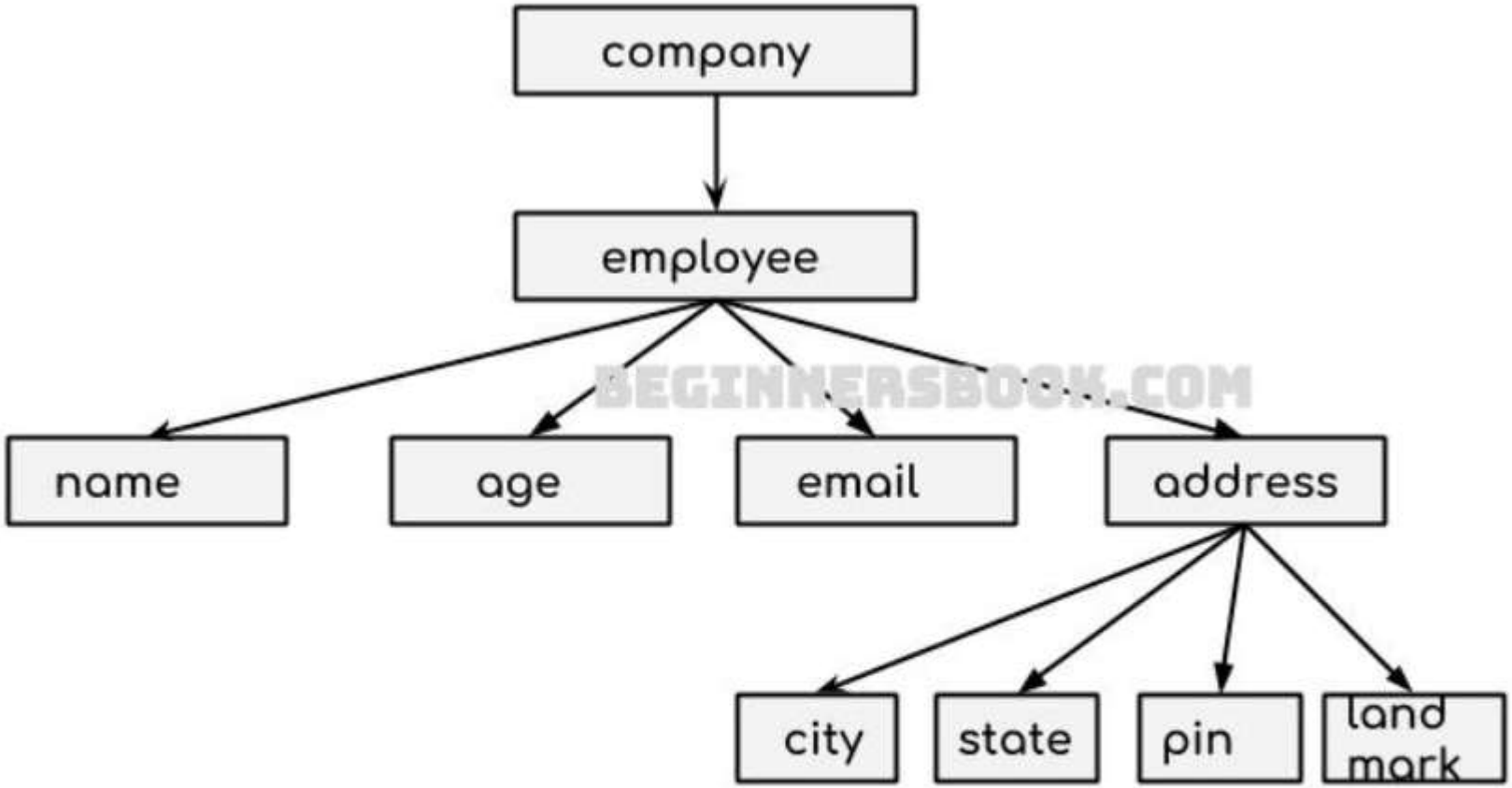
11:44 AM
03-Aug-16

XML Tree Structure

- XML documents have a hierarchical structure and can conceptually be interpreted as a tree structure, called an **XML tree**.

Example of XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<company>
  <employee>
    <name>Negan</name>
    <age>40</age>
    <email>imnegan@twd.com</email>
    <address>
      <city>Noida</city>
      <state>Uttar Pradesh</state>
      <pin>201301</pin>
      <landmark>Near hill top</landmark>
    </address>
  </employee>
</company>
```



Root element: `<company>`

Child elements: `<name>`, `<age>`, `<email>` & `<address>`

Sub-child elements: `<city>`, `<state>`, `<pin>` & `<landmark>`

HTML	XML
Hyper Text Markup Language	eXtensible Markup Language
Designed to **present/display** data	Designed to **Store and Transport** Data
Focus is on **How Data looks**	Focus is on What data is i.e **Describes the structure of a document**
Not Case Sensitive	**Case Sensitive**
Predefined Tags	**User defined tags**
Not required to close each and every tag Eg: <img><input>	All XML elements must have a Closing tag
Attribute values may or may not be quoted	Attribute values need to be quoted
Root element is <HTML>	Root element is user-defined
Saved as .html	Saved as .xml

Advantages of XML

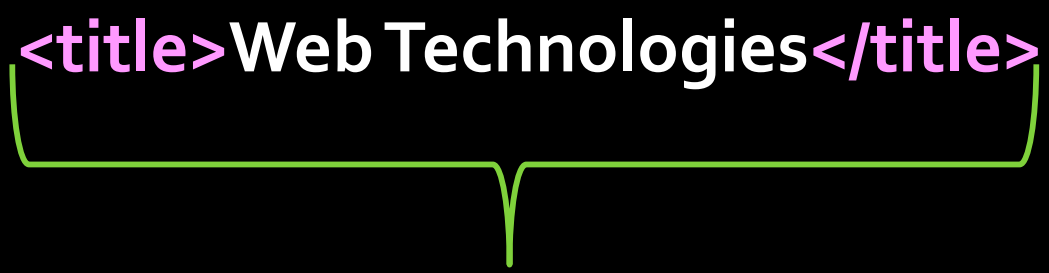
- Separates Data from HTML
- Simplifies Data Sharing and Transport – Platform Independent
- Simplifies Platform Changes – changes in hardware/software will not effect XML files
- Makes your data more available
- Readability – hierarchical structure
- Language Neutral

BASIC BUILDING BLOCKS

- XML documents are composed of 3 things:
- **1. XML Elements**
  - **Nested Tags**
  - **Case Sensitive**
  - **Empty Tags**
  - **Attributes**
- **2. Control Information**
  - **Comments**
  - **Processing Instructions**
  - **DTD**
- **3. Entities**

1. XML ELEMENTS

- Basic entity is 'element'
- Element – "**Everything from start tag to end tag"**
- Example:



- Each document has single root element which contains the other elements

- **1.1 Nested Tags:**
  - Tags inside Tags
  - Should be properly nested and closed in reverse order in which they were opened

  Example:

  ```
  <catalogue>
    <book>
          ----
    </book>
  </catalogue>
  ```

- **1.2 Case Sensitive:**
  - Generally, Lowercase letters are used for tags
  - Example: **<author>** **&** **<Author>** are different
- **1.3 Empty tags/elements:**
  - Elements without content
  - Place '/' before closing angle bracket
  - **Example:** **<title />**

1.4 Attributes: **<book id="101" > …… </book>**

2. CONTROL INFORMATION

- **2.1 XML Comments:**
  - <!-- Comment Text -- >
  - Example: **<!-- XML Introduction -->**
- **2.2 Processing Instructions:**
  - Used to pass information to applications
  - Need to be processed
  - **Syntax: <?target instruction ?>**
  - **Example: <?xml-stylesheet href = "mySheet.css" type = "text/css"?>**

2.3 Document Type Definition:

- Defines the basic building blocks of an XML file

3. ENTITIES

- Some characters have a special meaning in XML.
- **Example:** <message>salary **<** 1000</message>
  - **<** generates an error because the parser interprets it as the start of a new element.
  - To avoid this error, replace the "<" character with an entity reference **<**
- **Variables used to define shortcuts to common text or complex data**
- **Example :**

  kalam "Avul Pakir Jainulabdeen Abdul Kalam"

  Entity Value

  <name> &kalam </name>
- 3 types of entities:
  - **Internal Entities**
  - **External Entities**
  - **Predefined Entities**

There are 5 pre-defined entity references in XML:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

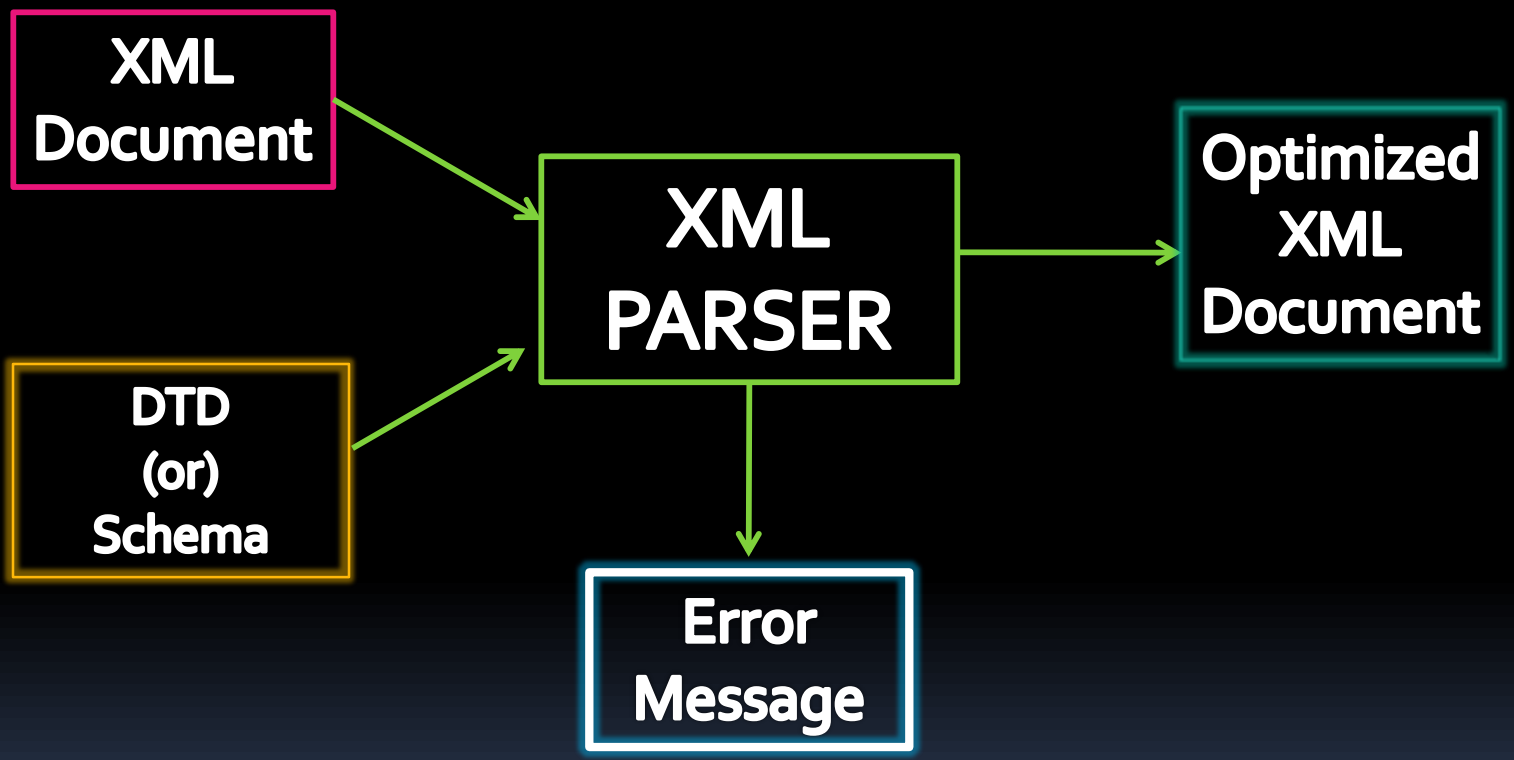
XML PARSERS

- Goal: to parse the given XML file
- Used to check: XML file
  - Well formed – syntax rules
  - Valid – internal rules defined in its DTD/Schema.

Types of Parsers

(1). Tree based - example DOM (Document Object Model)

(2). Event based – example SAX (Simple Api for XML)


XML Document
DTD (or) Schema
XML PARSER
Optimized XML Document
Error Message

Document Type Definition

- **DBMS Example:**

```
CREATE TABLE employees (
 emp_id INT NOT NULL,
 first_name VARCHAR(14)  NOT NULL,
 last_name VARCHAR(16) NOT NULL,
 department VARCHAR(30),
 PRIMARY KEY (emp_id)  );
```

- Defines the **basic building blocks of an XML** document
- **Example:** Elements, attributes, entities, order of elements and their occurrence
- **Grammar against which XML document is to be validated.**
- **Defines rules used by PARSERS to check that whether XML file is**
  - **Well –formed XML file** – syntax rules of XML
  - **Valid XML file** – Obeys rules defined in DTD/Schema

DATA TYPES IN DTD

- **PCDATA:**
  - Parsed Character DATA
  - Used with Elements
  - Element contains text that is going to be parsed by the XML parser - entities are replaced with values
- **CDATA:**
  - Character DATA
  - Used with Attributes
  - Plain text character data that is not parsed by the XML parser

DTD Syntax : ELEMENTS

General Syntax:

<!ELEMENT element-name (content) >

or

<!ELEMENT element-name (#data-type)>

- **Empty Element: without text**
  - Syntax: **<!ELEMENT element-name EMPTY>**
  - Example:

    <!ELEMENT br EMPTY>

- **SIMPLE ELEMENTS: only text**
  - Syntax: **<!ELEMENT element-name (#PCDATA) >**
  - Example: <title> This is a title </title>

    <!ELEMENT title (#PCDATA) >
- **COMPOUND ELEMENTS: child elements**
  - Syntax:

    <!ELEMENT element-name (child1, child2, -- - -)>
  - Example:

    <employee>

    <name>XXX</name>

    <id>123</id>

    <dept>CSE</dept>

    </employee>

    <!ELEMENT employee (name, id, dept)>

DTD SYNTAX : ATTRIBUTES

- (name,value) pairs
- Add additional information about element's content
- **Syntax:**
  - **<!ATTLIST element-name attribute-name CDATA attribute-value) >**
- **Attribute values:**
  - **Value - default value**
  - **#REQUIRED - mandatory attribute**
  - **#IMPLIED – may or may not be present**
  - **#FIXED value - value cannot be changed**

- Example:
  - <price currency="INR"> 100 </price>
  - <!ATTLIST price currency CDATA #REQUIRED>
- **DTD EXAMPLE:**

```
<?xml version="1.0" ?>
<students-list>
   <student id="1">
      <name></name>
      <address></address>
      <std>3</std>
      <marks>70</marks>
   </student>
</students-list>
```

TYPES OF DTD

- **INTERNAL DTD:**

```
<?xml version="1.0"> ?>
<!DOCTYPE   rootelement  [
            <! - - Rest of DTD  -->
 ]>
<rootelement>
  ……
</rootelement>
```

- **EXTERNAL DTD:**
  - An External DTD file is created and is saved with an extension .dtd
  - Syntax: <!DOCTYPE rootelement SYSTEM "URL">

PUBLIC

OCCURRENCE INDICATORS

- * - Zero or more times
- + - one or more times
- None - only once
- ? - zero or once
- | - set of alternatives. Only one may appear

Disadvantages of DTDs:

- Doesn't support **different data types** – It supports only the *text* ***string data type.***
- **Syntax is different from XML**
- Doesn't support **namespaces –**
  - Namespace is a **mechanism by which element and attribute names can be assigned to groups**.
  - **Used to resolve naming conflicts**

XML Schemas

- **It is used to describe and validate the structure and the content of XML data.**
- Defines the **basic building blocks of an XML** document
- XML schema **defines the elements, attributes and data types.**
- Allows to specify **different data types for elements**.
- Schema element supports **Namespaces**.
- XML Schema is commonly known as **XML Schema Definition (XSD)** are saved with an extension **.xsd**

- **Syntax :**

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

XML Schema Data types

- **String**
- **Date** –"yyyy – mm – dd"
- **Numeric (decimal, short, int, integer, float, long, double)**
- **Boolean**
- **Time** – "hh : mm : ss"

- **Syntax:**
  - **<xs:element name="xxx" type="xs:datatype" />**

ELEMENTS

- Simple Type – contains only text
  - <xs:element name="xxx" type="xs:datatype" />
- Complex Type – contains attributes and/or child elements

```
<xs:element name="xxx">
  <xs:complexType>
    <xs:sequence>
          <! – Child Elements -->
    </xs:sequence>
        <! – Attributes  -->
  </xs:complexType>
</xs:element>
```

Attributes

- **Syntax:**
  - <xs:attribute name = “xxx” type=“xs:datatype />
- **Values for attributes:**
  - Default
    - <xs:attribute name = “xxx” type=“xs:datatype" default= “DefaultValue” />
  - **Fixed**
    - <xs:attribute name = “xxx” type=“xs:datatype" fixed= “FixedValue” />
  - **Optional**
  - **Required**
    - <xs:attribute name = “xxx” type=“xs:datatype use= “required”/>

Including XML Schema in XML:

- <rootelement xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation=".xsd">