eXtensible Markup Language (XML)

Dr.S.Nagender Kumar
School of Computer and Information Sciences
University of Hyderabad

Outline

- XML basics
- XML syntax rules
- XML Namespaces
- Document Type Definition
- XML Schema
- XML Parsers
- Relevant demos

XML

- Computers contain data in incompatible formats
  - They need some means to exchange the data
  - Robust, extensible, standard format for data
- XML is mainly designed to store and transport data
- It is self-descriptive and both human and machine readable
- Mostly information wrapped in tags
  - Similar to HTML
- However, XML and HTML design goals are different
  - XML is designed to carry the data, while HTML is for displaying the data
  - HTML tags are predefined but XML tags are not
- It is extensible
  - Mostly designed so that applications still work even when there will be some/minor modifications in data
- XML is a W3C Recommendation from 1998

XML Document - Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>33.25</price>
  </book>
  <book category="programming">
    <title lang="en">Learning Python</title>
    <author>Mark Lutz</author>
    <year>2015</year>
    <price>44.94</price>
  </book>
  <book category="web">
    <title lang="en">Java Servlet Programming</title>
    <author> William Crawford and Jason Hunter</author>
    <year>2018</year>
    <price>15.71</price>
  </book>
</bookstore>
```

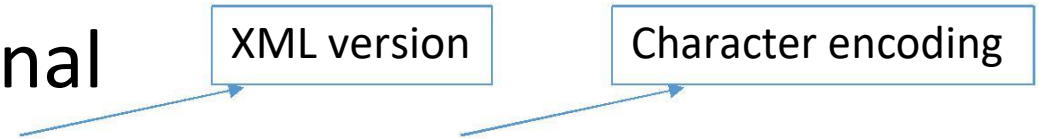
Tree Structure

- XML documents are formed as element trees
- Starts with a root element
- Followed by child elements
- Elements can have sub elements
- Relationships between elements
  - Parent
  - Child
  - Sibling
- XML documents should follow XML syntax rules

XML Syntax rules

- XML documents must have a root element
- XML prolog is optional
  - <?xml version="1.0" encoding="UTF-8"?>
  - Should be the first line, if it exists
  - Actually not part of XML document
- All XML elements must have a closing tag

```
<author></author>
<br />
```



XML Syntax rules - continued

Demo 1

- XML tags are case sensitive
  - Both opening and closing tags should have same case
- XML elements must be nested properly
- XML attribute values must always be quoted
- "Well Formed" XML documents
  - XML documents that conform to these syntax rules

XML Syntax rules - Entity references

- White space
  - White spaces are preserved in XML
  - Other white space characters in XML
    - Carriage return, line feed, and tab

- & - `&`
- < - `<`
- > - `>`
- ‘ - `&apos;`
- “ - `&quot;`

XML Syntax rules - continued

- XML comments

```
<!-- This is a comment -->
```

- Character data
  - CDATA section
    - The entire content of this is interpreted as character data
    - Can contain markup data
    - No need to maintain entity references
    - Placed between <![CDATA[AND]]>

```
<![CDATA[
    <message>
    This can contain anything
    </message>
]]>
```

XML Elements

- XML element is everything between the start tag and the end tag, including the tags
- Element can contain
  - text, attributes, other elements
- Empty elements contain no content
  - <element></element> - can also be represented as <element />
  - Empty elements can have attributes

XML Elements - continued

- XML naming rules
  - Must start with a letter or underscore
  - Cannot start with the letters xml
  - Can contain letters, digits, hyphens, underscores, and periods
  - Cannot contain spaces
- XML elements are extensible
  - Elements can be extended to contain more information
    - E.g. <book> can be extended to contain <description> element
  - Applications should not crash
    - Should still be able to extract the current information

https://www.w3schools.com/xml/xml_elements.asp

XML Attributes

- XML elements can have attributes
- Attributes are good for metadata
  - e.g. id in a tag
- Avoid attributes to the extent possible
  - Instead opt for elements
  - attributes cannot contain multiple values
  - attributes cannot contain tree structures
  - attributes are not easily expandable

XML Validation

Syntax | Structure

Is well formed (satisfy all the basic syntax condition)

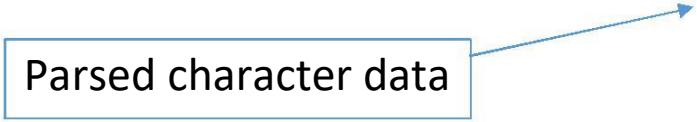
Behaves according to predefined DTD (Document Type Definition)

Or

XSD (XML Schema Definition)

DTD (Document Type Definition)

- DTD defines the structure of the XML document
  - Defines legal elements and attributes
- Applications can verify the validity of XML through DTD
  - So people can agree on a standard for exchanging the data
- From a DTD point of view XML documents contain
  - Elements, Attributes, Entities, PCDATA and CDATA



Internal DTD declaration and example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bookstore [
<!ELEMENT bookstore (book+)>
<!ELEMENT book (title,author,year,price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST book category (web|programming) "web">
<!ATTLIST title lang CDATA #REQUIRED>
]>
<bookstore>
  <book category="web">
```

- External DTD declaration
  - <!DOCTYPE bookstore SYSTEM “bookstore.dtd">

DTD Elements

- Declaring elements

  <!ELEMENT element-name (element-content)>
  <!ELEMENT bookstore (book)>
- Empty element

  <!ELEMENT br EMPTY>
- Elements with parsed character data

  <!ELEMENT title (#PCDATA)>
- Elements with one or more child elements

  <!ELEMENT book (title,author,year,price)>
- Declaring multiple occurrences

  <!ELEMENT bookstore (book)> - only one occurrence
  <!ELEMENT bookstore (book+)> - Minimum once
  <!ELEMENT bookstore (book*)> - Zero or more times
  <!ELEMENT bookstore (book?)> - Zero or once

https://www.w3schools.com/xml/xml_dtd_elements.asp

DTD Attributes

- Declaring attributes

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

  <!ATTLIST book category CDATA "web">

- If attribute is required

  <!ATTLIST title lang CDATA #REQUIRED>

  – #FIXED – value is fixed

  – #IMPLIED – do not want to force a value

- Enumerated attribute values

  <!ATTLIST book category (web|programming) "web">

DTD - Entities

Demo 2

- Internal entity declaration

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

  <!ENTITY writer “">

  <author>&writer;</author> - usage in XML

- External entity declaration
  - <!ENTITY entity-name SYSTEM "URI/URL">
  - <!ENTITY writer SYSTEM "https://uohyd.ac.in/entities.dtd">

Schema

- XML Schema is XML-based alternative to DTD
  - To define the structure of the XML document
- You don't have to learn a new language
- Schemas also support data types
  - It is easier to describe document content
  - It is easier to define restrictions on data
  - It is easier to validate the correctness of data

XML Schema example

- Referencing schema in a document

```
<bookstore xmlns="MyNameSpace"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="bookstore.xsd">
  <book category="web">
```

- Defining simple elements in schema

```
<xs:element name="xxx" type="yyy"/>
<xs:element type="xs:float" name="price"/>
```

- There are several built in data types
  - xs:string, xs:decimal, xs:integer, xs:Boolean, xs:date, xs:time
- You can also set default="" or fixed="" to the element

XSD Attributes

- Syntax of defining an attribute

  <xs:attribute name="xxx" type="yyy"/>

  <xs:attribute type="xs:string" name="lang" default="en“ use="optional"/>

- Attributes are optional by default
  - use="required“ - makes them required
- Similar to element you can set default=“” or fixed=“”

XSD Complex elements

- If an element contains child elements or attributes, it becomes complex

```
<xs:element name="rectangle">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="point" minOccurs="1" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="x" type="xs:int" />
                    <xs:attribute name="y" type="xs:int" />
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="colour" type="xs:string" />
    </xs:complexType>
</xs:element>
```

XSD Restrictions/Facets

Demo 3

- Used to define acceptable values for XML elements or attributes

```
<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female|other"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{10}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

https://www.w3schools.com/xml/schema_facets.asp

XML Namespaces

- Provide a means to avoid element name conflicts
  - Conflicts occur when documents from multiple users are joined

```
<table>
  <tr>
    <th>Student id</th>
    <th>Quiz 1</th>
    <th>Quiz 2</th>
    <th>Quiz 3</th>
  </tr>
</table>
```

```
<table>
  <material>Wood</material>
  <width>90</width>
  <length>150</length>
</table>
```

- Name conflicts can be avoided using a name prefix

XML Namespaces - continued

- When using a prefix, the namespace for the prefix must be defined
  - xmlns:prefix="URI"
  - <h:table xmlns:h="http://www.w3.org/TR/html4/">
- Namespace defined for an element, is also associated for all its child elements with the same prefix
- You can also define namespaces in root element

```
<root>

<h:table xmlns:h=
"http://www.w3.org/TR/html4/">
  <h:tr>
    <h:th>Student id</h:th>
    <h:th>Quiz 1</h:th>
    <h:th>Quiz 2</h:th>
    <h:th>Quiz 3</h:th>
  </h:tr>
</h:table>

<f:table xmlns:f=
"https://www.w3schools.com/furniture">
  <f:material>Wood</f:material>
  <f:width>90</f:width>
  <f:length>150</f:length>
</f:table>

</root>
```

XML Parser

- Parser is a software library that reads the XML document and checks it for well-formedness
- Client applications use method calls defined in the parser API
  - to receive, request or modify information in the XML document
- Different types are parsers
  - DOM (Document Object Model)
  - SAX (Simple API for XML)
    - Event driven API

http://www.cafeconleche.org/books/xmljava/chapters/ch05s02.html

Parsers - DOM

- HTML DOM defines a standard way for accessing and manipulating HTML documents
- XML DOM defines a standard way for accessing and manipulating XML documents
- DOM models XML as a set of node objects
  - The entire document is a document node
  - Every XML element is an element node
  - The text in the XML elements are text nodes
  - Every attribute is an attribute node
  - Comments are comment nodes
- The nodes can be accessed with JavaScript or other programming languages

Working with DOM Demo 4

- Getting access to DOM parser from JavaScript

```
parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");
```

- Accessing nodes
  - By using the getElementsByTagName() method
  - By traversing the node tree
  - By navigating the node tree, using node relationships
- documentElement property of the XML document is the root node

https://www.w3schools.com/xml/dom_intro.asp

References

- XML tutorial
  https://www.w3schools.com/xml/default.asp
- XML Schema tutorial

https://www.liquid-technologies.com/xml-schema-tutorial/xsd-elements-attributes