XML, DTD, XML Schema, XSLT

What is XML?

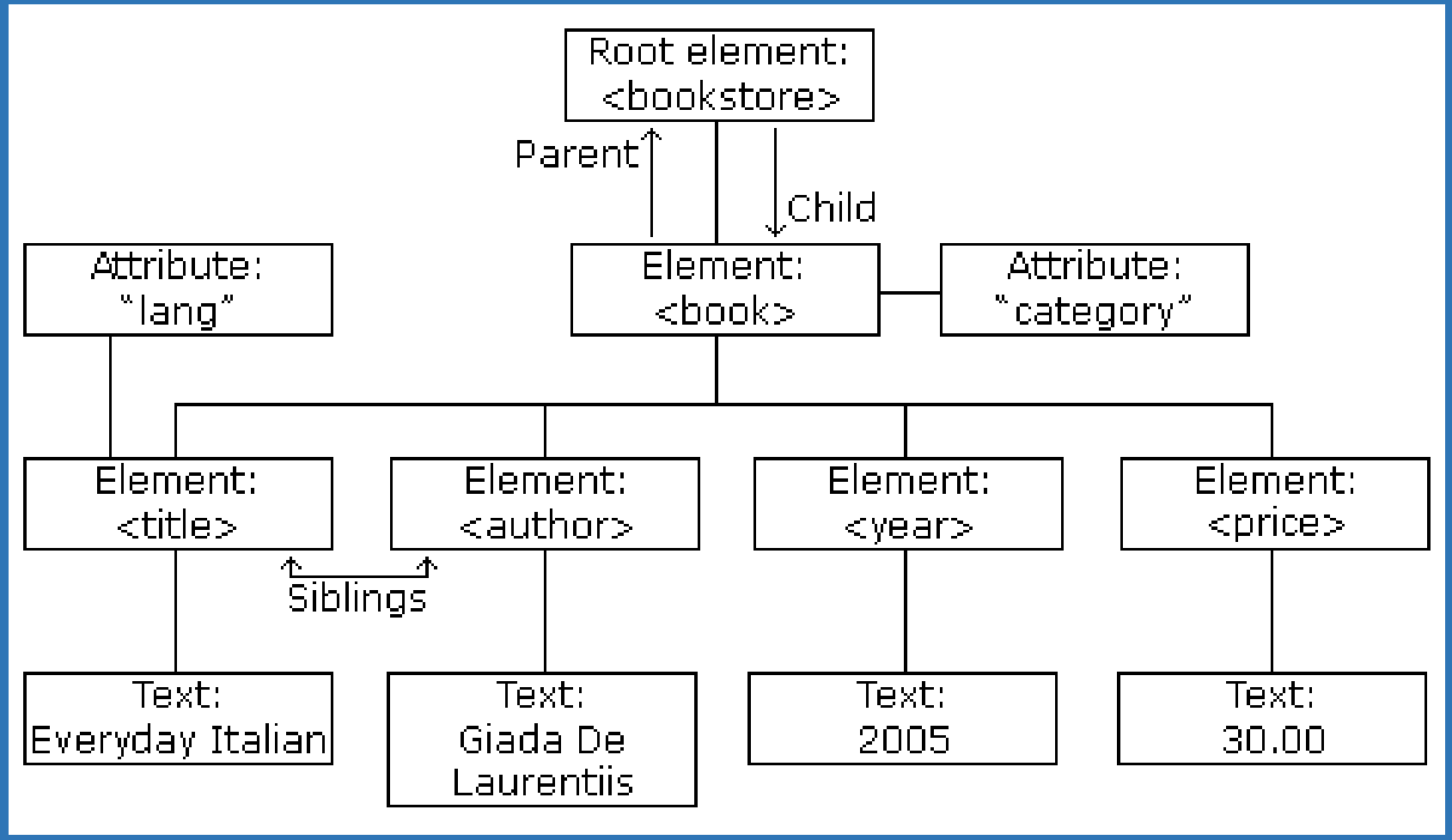
"XML stands for **E**xtensible **M**arkup **L**anguage. It is a text-based markup language derived from Standard Generalized Markup Language (SGML)."

- XML stands for Extensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML Usage

- XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange the information between organizations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Virtually, any type of data can be expressed as an XML document.

XML Tree Structure



Example

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```

XML with CSS

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Empire</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
</CATALOG>
```

cd_catalog.css

```
CATALOG {
    background-color: #ffffff;
    width: 100%;
}
CD {
     margin-bottom: 30pt;
    margin-left: 0;
}
TITLE {
    color: #ff0000;
    font-size: 20pt;
}
ARTIST {
    color: #0000ff;
    font-size: 20pt;
}
COUNTRY, PRICE, YEAR, COMPANY {
    color: #000000;
    margin-left: 20pt;
}
```

What is DTD?

- The XML **Document Type Definition** , commonly known as DTD, is a way to describe XML language precisely.
- DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language.

DTD Syntax

```
<!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 *Internal Subset*.

Internal DTD Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE address [
   <!ELEMENT address (name,company,phone)>
   <!ELEMENT name (#PCDATA)>
   <!ELEMENT company (#PCDATA)>
   <!ELEMENT phone (#PCDATA)>
]>
<address>
   <name>Shravan Kumaran</name>
   <company>Go Dimension</company>
   <phone>(011) 123-4567</phone>
</address>
```

External DTD Example

```
?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

DTD - Elements

- Declaring Elements

```
<!ELEMENT element-name category>
or
<!ELEMENT element-name (element-content)>
```

- Empty Elements

<!ELEMENT element-name EMPTY>

Example:

<!ELEMENT br EMPTY>

XML example:

DTD - Elements

- Elements with Parsed Character Data

  Example:

  ```
  <!ELEMENT from (#PCDATA)>
  ```

- Elements with any Contents

  Example:

  ```
  <!ELEMENT note ANY>
  ```

- Elements with Children (sequences)

  ```
  <!ELEMENT note (to,from,heading,body)>
  ```

DTD - Elements

- Declaring Only One Occurrence of an Element

```
<!ELEMENT note (message)>
```

- Declaring Minimum One Occurrence of an Element

```
<!ELEMENT note (message+)>
```

- Declaring Zero or More Occurrences of an Element

```
<!ELEMENT note (message*)>
```

- Declaring Zero or One Occurrences of an Element

```
<!ELEMENT note (message?)>
```

- Declaring either/or Content

```
<!ELEMENT note (to,from,header,(message|body))>
```

DTD - Attributes

- A Default Attribute Value

```
DTD:
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">

Valid XML:
<square width="100" />
```

- #REQUIRED

```
DTD:
<!ATTLIST person number CDATA #REQUIRED>

Valid XML:
<person number="5677" />

Invalid XML:
<person />
```

DTD - Attributes

- #IMPLIED

  DTD:
  <!ATTLIST contact fax CDATA #IMPLIED>

  Valid XML:
  <contact fax="555-667788" />

  Valid XML:
  <contact />
- #FIXED

  DTD:
  <!ATTLIST sender company CDATA #FIXED "Microsoft">

  Valid XML:
  <sender company="Microsoft" />

  Invalid XML:
  <sender company="Apple" />

Example

```
<?xml version="1.0"?>
<!DOCTYPE   shiporder[
<!ELEMENT   shiporder   (orderperson,shipto,item+)>
<!ATTLIST   shiporder orderid CDATA #REQUIRED>
<!ELEMENT   orderperson (#PCDATA)>
<!ELEMENT   shipto(name,address,city,country)>
<!ELEMENT   name  (#PCDATA)>
<!ELEMENT   address(#PCDATA)>
<!ELEMENT   city  (#PCDATA)>
<!ELEMENT   country(#PCDATA)>
<!ELEMENT   item  (title,note?,quantity, price)>
<!ELEMENT   title (#PCDATA)>
<!ELEMENT   note  (#PCDATA)>
<!ELEMENT   quantity(#PCDATA)>
<!ELEMENT   price (#PCDATA)>
]>
```

Solution

```
<?xml version="1.0"?>
<shiporder orderid="889923">
      <orderperson>John Smith </orderperson>
      <shipto>
            <name>      Ola   Nordmann    </name>
            <address>   Langgt      23     </address>
            <city>      4000  Stavanger    </city>
            <country>   Norway       </country>
      </shipto>
      <item>
            <title>     Empire      Burlesque</title>
            <note>      Special     Edi/on      </note>
            <quantity>  1     </quantity>
            <price>     10.90 </price>
      </item>
      <item>……</item>
</shiporder>
```

Exercise

```
<!DOCTYPE NEWSPAPER [
<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>
```

CDATA vs PCDATA

CDATA: (Unparsed Character data): CDATA contains the text which is not parsed further in an XML document. Tags inside the CDATA text are not treated as markup and entities will not be expanded.

Let's take an example for CDATA:

```
<?xml version="1.0"?>
<!DOCTYPE employee SYSTEM "employee.dtd">
<employee>
<![CDATA[
  <firstname>xyz</firstname>
  <lastname>pqr</lastname>
  <email>xyz@pqr.com</email>
]]>
</employee>
```

Cont…

PCDATA: (Parsed Character Data): XML parsers are used to parse all the text in an XML document. PCDATA stands for Parsed Character data. PCDATA is the text that will be parsed by a parser. Tags inside the PCDATA will be treated as markup and entities will be expanded.

Let's take an example:

```
<?xml version="1.0"?>
<!DOCTYPE employee SYSTEM "employee.dtd">
<employee>
  <firstname>xyz</firstname>
  <lastname>pqr</lastname>
  <email>xyz@pqr.com</email>
</employee>
```

What is XML Schema?

- 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.
- It is similar to a database schema that describes the data in a database.

Example (Filename.xsd)

```
<?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>
```

Defining a Simple Element

A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.

XML Schema has a lot of built-in data types. The most common types are:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Example

XML Element:

```
<lastname>PQR</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

And here are the corresponding simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Cont…

- Default and Fixed Values for Simple Elements

A default value is automatically assigned to the element when no other value is specified.

```
<xs:element name="color" type="xs:string" default="red"/>
```

A fixed value is also automatically assigned to the element, and you cannot specify another value.

```
<xs:element name="color" type="xs:string" fixed="red“  use="required"/>
```

Restriction on Content

- Restrictions on a Value

```
<xs:element name="age">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value="120"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Cont…

- Restriction on set of Value

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value=“Maruti"/>
      <xs:enumeration value=“TATA"/>
      <xs:enumeration value=“FORD"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Cont…

- Restrictions on a Series of Values

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Cont…

The acceptable value is zero or more occurrences of lowercase letters from a to z:

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Cont…

The next example also defines an element called "letter" with a restriction. The acceptable value is one or more pairs of letters, each pair consisting of a lower case letter followed by an upper case letter. For example, "sToP" will be validated by this pattern, but not "Stop" or "STOP" or "stop":

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z][A-Z])+"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Cont…

- Restrictions on Whitespace Characters

To specify how whitespace characters should be handled, we would use the whiteSpace constraint.

The whiteSpace constraint is set to "preserve", which means that the XML processor WILL NOT remove any white space characters:

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Cont…

- Restrictions on Whitespace Characters

The whiteSpace constraint is set to "replace", which means that the XML processor WILL REPLACE all white space characters (line feeds, tabs, spaces, and carriage returns) with spaces:

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="replace"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Cont…

- Restrictions on Whitespace Characters

The whiteSpace constraint is set to "collapse", which means that the XML processor WILL REMOVE all white space characters (line feeds, tabs, spaces, carriage returns are replaced with spaces, leading and trailing spaces are removed, and multiple spaces are reduced to a single space):

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="collapse"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Cont…

- Restrictions on Length

To limit the length of a value in an element, we would use the length, maxLength, and minLength constraints.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions for Datatypes

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whitespace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled