XML SCHEMA TYPES AND DTD TYPES

SIMPLE TYPE:

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:

Here are some XML elements:

<lastname>Refsnes</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"/>

COMPLEX TYPE:

A complex element is an XML element that contains other elements and/or attributes. Look at this simple XML document called "note.xml":

<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget to submit the assignment this monday!</body>
</note>

The following example is a DTD file called "note.dtd" that defines the elements of the XML document above ("note.xml"):

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

The following example is an XML Schema file called "note.xsd" that defines the elements of the XML document above ("note.xml"):

<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com" elementFormDefault="qualified">
<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>
```

DTD:

A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes.

TWO TYPES OF DTD

- INTERNAL DTD
- EXTERNAL DTD

INTERNAL DTD:

If the DTD is declared inside the XML file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element [element-declarations]>
Example XML document with an internal DTD:
<?xml version="1.0"?>
<!DOCTYPE note [ <!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)> ]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget to prepare for the UNIT TEST this weekend</body></note>
```

EXTERNAL DTD:

If the DTD is declared in an external file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element SYSTEM "filename">
This is the same XML document as above, but with an external DTD
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget to prepare for the UNIT TEST this weekend!</body>
</note>
And this is the file "note.dtd" which contains the DTD:
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>


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

<!ELEMENT body (#PCDATA)>

XML SCHEMAS:

XML Schema is an XML-based alternative to DTDs. An XML Schema describes the structure of an XML document. The XML Schema language is also referred to as XML Schema Definition (XSD).The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.

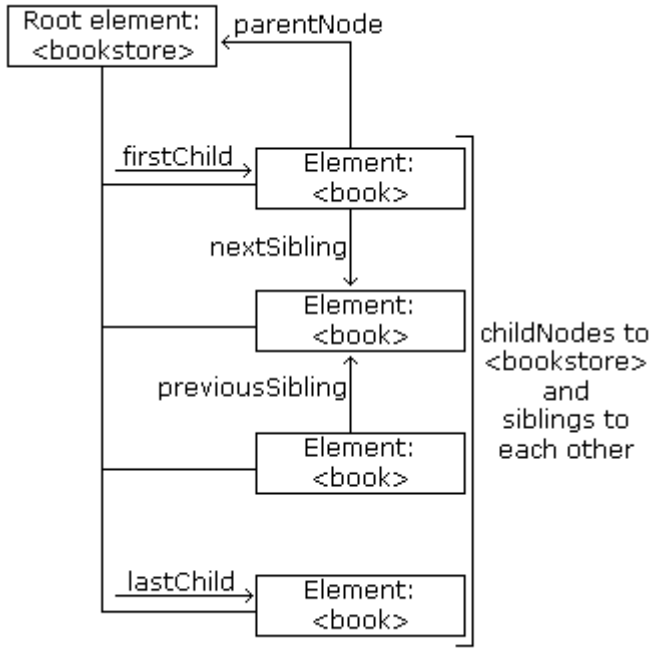
An XML Schema defines elements that can appear in a document, defines attributes that can appear in a document, defines which elements are child elements , defines the order of child elements, defines the number of child elements, defines whether an element is empty or can include text, defines data types for elements and attributes, defines default and fixed values for elements and attributes.

XML PROCESSING:

The JavaTM API for XML Processing (JAXP) includes the basic facilities for working with XML documents through the following standardized set of Java Platform APIs. There are two types of XML Parsers namely Document Object Model (DOM), Simple API For XML Parsing (SAX).

DOM:

The XML DOM views an XML document as a tree-structure. The tree structure is called a node-tree. All nodes can be accessed through the tree. Their contents can be modified or deleted, and new elements can be created. The nodes in the node tree have a hierarchical relationship to each other. The terms parent, child, and sibling are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings (brothers or sisters). In a node tree, the top node is called the root. Every node, except the root, has exactly one parent node. A node can have any number of children. A leaf is a node with no children. Siblings are nodes with the same parent.



SAX:

SAX (Simple API for XML) is an event-driven model for processing XML. Most XML processing models (for example: DOM and XPath) build an internal, tree-shaped

representation of the XML document. The developer then uses that model's API (getElementsByTagName in the case of the DOM or findnodes using XPath, for example) to access the contents of the document tree. The SAX model is quite different. Rather than building a complete representation of the document, a SAX parser fires off a series of events as it reads the document from beginning to end. Those events are passed to event handlers, which provide access to the contents of the document.

Source : http://nprcet.org/e%20content/Misc/e-Learning/IT/VIII%20Sem/IT1451%20-%20XML%20and%20Web%20Services.pdf