

Workbook for XML

A beginner's guide to effective programming

Why This Module

As many people predicted, XML has changed the world! When XML was introduced, many considered it a revolutionary leap forward in how we look at data representation. Since then, XML has grown considerably and many new technologies have been introduced. The number of new technologies based upon XML is staggering. XML is simply the most robust, reliable, and flexible document syntax ever invented.












This module covers the basics of XML. This module focuses on Microsoft technology and hence Microsoft Visual Studio is used to enhance the examples. It covers all aspects of XML, from the most basic syntax rules, to the details of DTD and schema creation, to the APIs you use to read and write XML documents in a variety of programming languages. It also includes search and manipulation technology like XQuery that often completes with XPath.

Contents

1. XML Concepts	6-24
1.1 Introduction to XML	7
1.2 XML Elements and Attributes	14
1.3 XML Namespaces	17
1.4 XML Tree	21
1.5 Crossword	24
2. XML Document	25-44
2.1 XML Document Structure and Syntax	26
2.2 DTD (Document Type Definition)	29
2.3 XSD (XML Schema Definition)	34
2.4 Crossword	43
3. XML Parsing Concepts	44-59
3.1 XML DOM (Document Object Model)	45
3.2 SAX and other XML Parsers	54
3.3 Crossword	57
4. XML Transformations	58-81
4.1 Introduction to XSLT	59
4.2 Building Blocks of XSLT	64
4.4 Crossword	81
5. Querying XML	82-98
5.1 XPath	83
5.2 XQuery	94
5.3 Crossword	98
**Answers For Crosswords	99
**Contributors	100

Guide to Use this Workbook

Conventions Used






Convention	Description
 Topic	Indicates the Topic Statement being discussed.
Estimated Time	Gives an idea of estimated time needed to understand the Topic and complete the Practice session.
 Presentation	Gives a brief introduction about the Topic.
 Scenario	Gives a real time situation in which the Topic is used.
 Demonstration/Code Snippet	Gives an implementation of the Topic along with Screenshots and real time code.
<i>Code in Italic</i>	Represents a few lines of code in Italics which is generated by the System related to that particular event.
// OR	Represents a few lines of code (Code Snippet) from the complete program which describes the Topic.
 Context	Explains when this Topic can be used in a particular Application.
 Practice Session	Gives a practice example for the participant to implement the Topic, which gives him a brief idea how to develop an Application using the Topic.
 Check list	Lists the brief contents of the Topic.
 Common Errors	Lists the common errors that occur while developing the Application.
 Exceptions	Lists the exceptions which result from the execution of the Application.
 Lessons Learnt	Lists the lessons learnt from the article of the workbook.
 Best Practices	Lists the best ways for the efficient development of the Application.

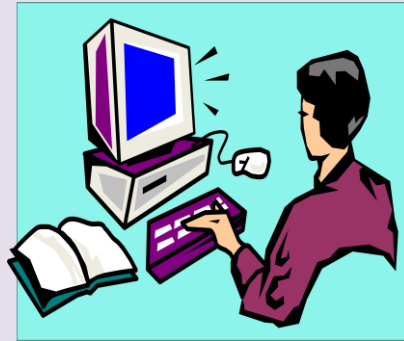
**Notes**

Gives important information related to the Topic in form of a note

1.0 XML Concepts

Topics

-  1.1 Introduction to XML
-  1.2 XML Elements and Attributes
-  1.3 XML Namespaces
-  1.4 XML Tree
-  1.5 Crossword



Topic: Introduction to XML

Estimated Time: 75 Mins.



Objectives: This module will familiarize the participant with

- Importance of XML than other markup languages
- XML document example
- Nature of XML
- Applications of XML



Presentation:

- XML stands for **Extensible Markup Language**
- XML is used to store and transfer data via any standard Internet Protocol.
- XML tags are user defined tags.
- XML is a W3C Recommendation.

Importance of XML than other markup languages

- XML is Extensible Markup Language because it does not have a fixed format like HTML.
- Designed for interoperability with both SGML (Standard Generalized Markup Language) and HTML.
- HTML is a markup language for presentation for the browser, whereas, XML is meant to represent and transfer data.

Nature of XML:

- A valid XML Document is Well-Formed.
- Well Formed is with respect to complete structure of the tags
- An XML document is valid if each and every tag is defined.
- XML is case sensitive.
- It must have one and only one root element.

Applications of XML:

- To Search data efficiently
- For Website management
- For Inter application Communication
- In E-Commerce application.



Scenario :

An example of XML document is “first.xml”, which contains root element `<satyam>`, child element `<eltp>` which represent training and `<stream>` element that represents different stream like .Net, Java, Oracle and Main Frame. That are offered for training.



Demonstration/Code Snippet :

In Visual studio 2005:

Step 1:

Create an XML file in visual studio 2005, file menu → new file → XML file → “first.xml”. the extension of the xml file is **.xml**.

In Notepad:

Step 1:

Click on “Start Menu”→”Run Menu”→type “Notepad”→press “Enter” key”. Next create an XML file and save it with extension xml.

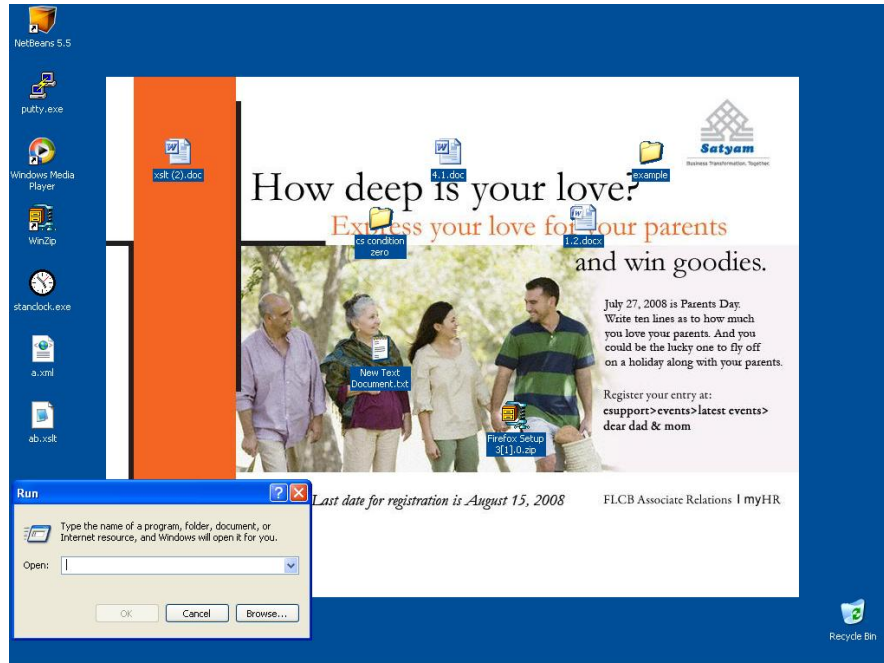


Figure 1.1-1: Opening Run from start menu



Figure 1.1-2: Opening Notepad

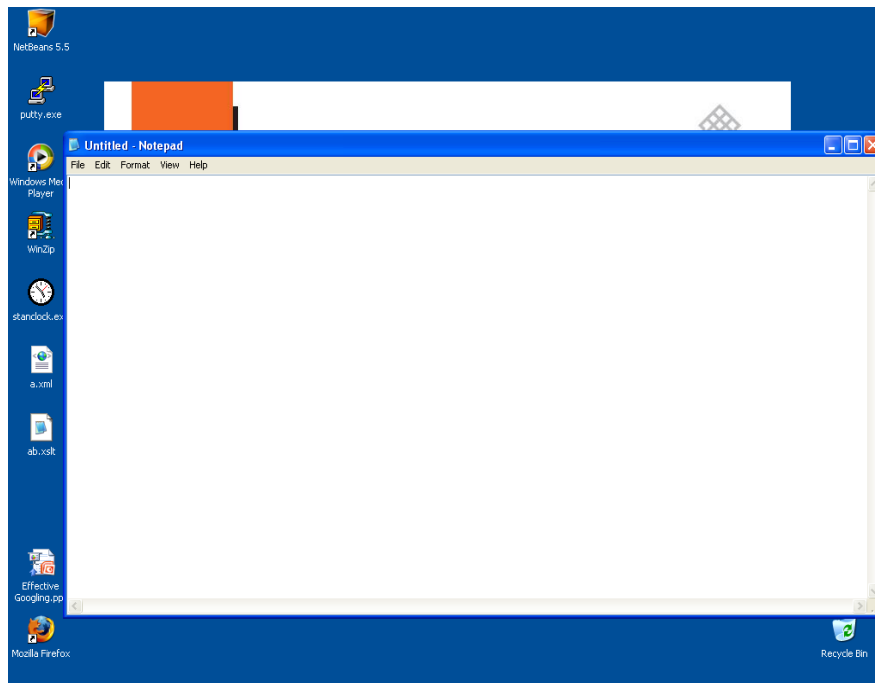


Figure 1.1-3: Notepad opened

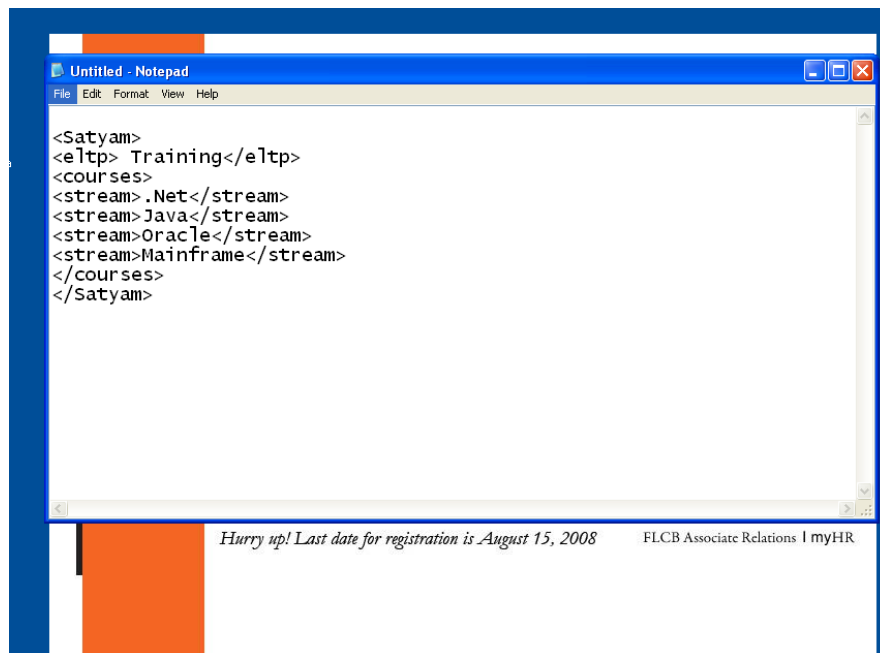


Figure 1.1-4: Example Demo

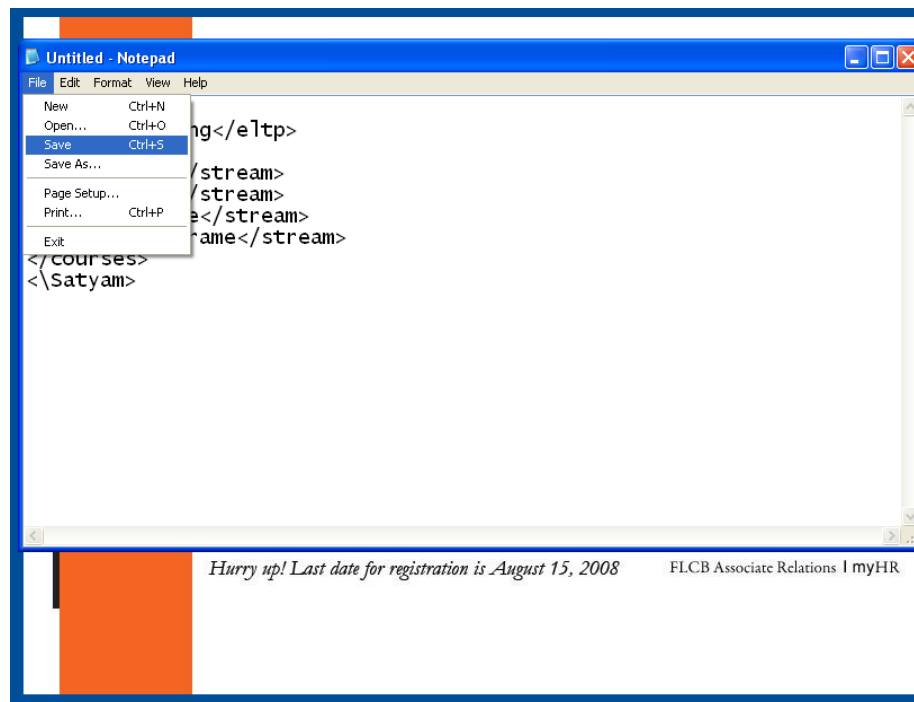


Figure 1.1-5: Saving the file

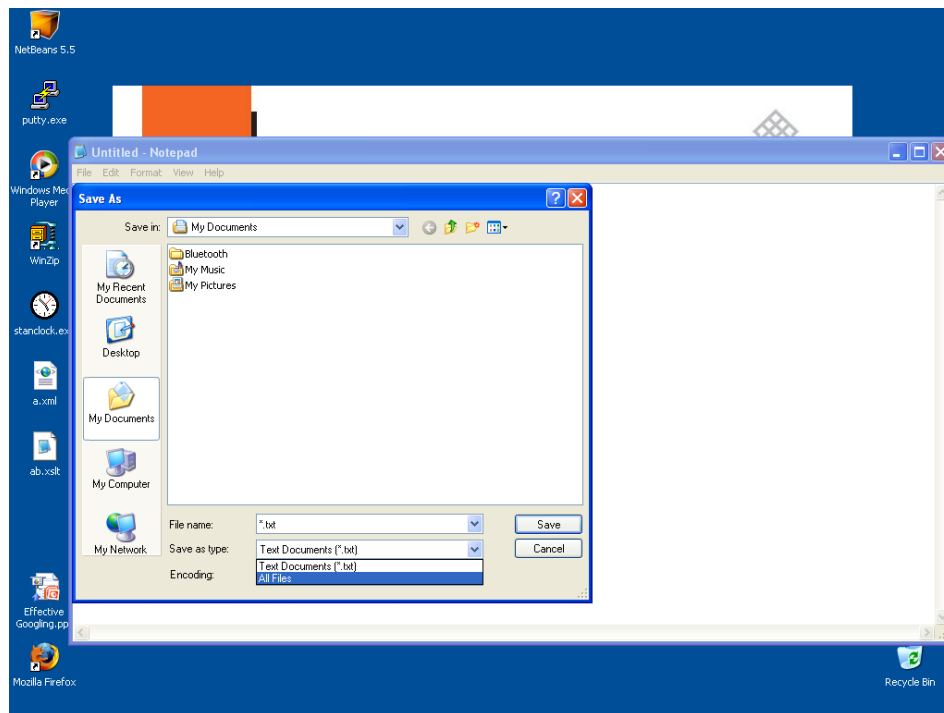


Figure 1.1-6: Saving the file

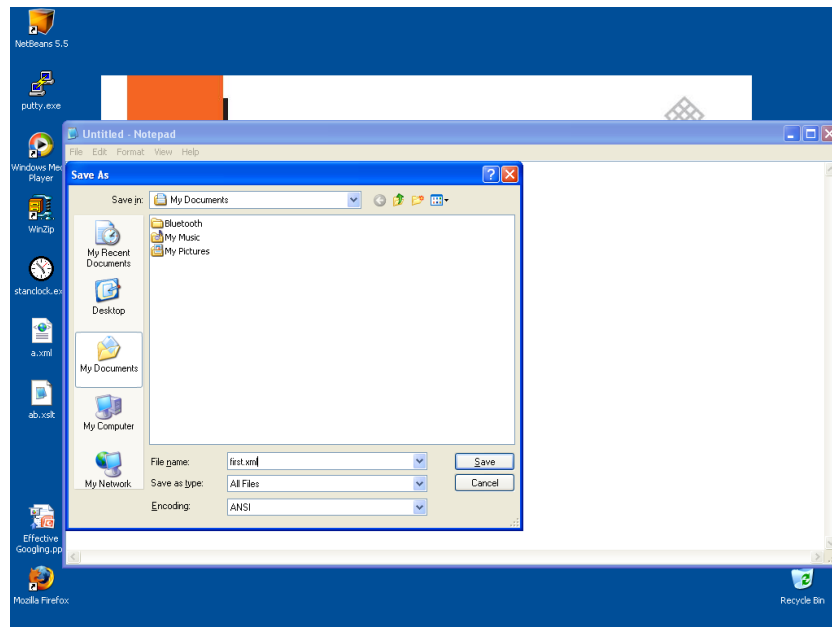


Figure 1.1-7: Saved file

```
<Satyam>

  <eltp> Training</eltp>

  <courses>

    <stream>.Net</stream>

    <stream>Java</stream>

    <stream>Oracle</stream>

    <stream>Mainframe</stream>

  </courses>

</Satyam>
```



Context :

- Introduction to XML and basics.



Practice Session :

- Write an XML document describing your course details with attributes like course name and subchild nodes like duration, credit and sub modules.



Check List :

- Introduction of XML and XML document.
- Nature and Applications of XML.



Common Errors :

- Every tag must have a closing tag.
- Multiple roots must be avoided.
- As XML is case sensitive, use same tags. For e.g do not use <width> and </WIDTH>



Lessons Learnt :

- ☒ How to write a simple XML document.



Best Practices :

- Make it a practice to write Valid and Well-Formed XML document.



Topic: XML Elements and Attributes

Estimated Time: 80 Mins.



Objectives : This module will familiarize the participant with

- XML elements
- Attributes of an element
- Text
- Empty elements
- Comments in an XML document
- Naming conventions for elements and attributes



Presentation :

XML Elements:

- Elements can be defined as `<element></element>`
Example: `<satyam> </satyam>`
- An element can contain other elements, simple text or a mixture of both.

Attributes of an element:

- Attributes contain values that are associated with an element.
- They are a part of elements opening tag.

- Attribute value must be defined within quotes.
- Attributes can be defined as `<element attribute="value"></element>`
Example: `<eltp_training phase="1"> </eltp_training>`

Text:

- It is written between the opening and closing tags of an element.
- Text can be defined as `<element attribute="value">text</element>`

Example: `<module duration="2 days"> Operating System </module>`

Empty elements:

- Elements without attributes and text.
Example: `<computer/>`

Comments in XML:

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

Naming conventions:

- Element names can contain letters, numbers and underscores.
- It cannot contain spaces, hyphens, colons and dots.
- It cannot start with non-alphabetical characters or numbers or the letters 'xml'.



Scenario :

The training for the new joiners in Satyam takes place in 2 phases. In 1st phase, the modules covered are C & DS, Operating System, RDBMS, Web Programming and Software Engineering. Prepare an XML document describing the duration of each module as well as for the entire phase.



Demonstration/Code Snippet :

Step 1: Create an XML file in Visual Studio 2005, File menu → New File → XML file → "Satyamtraining.xml". Here extension of the xml file is **.xml**.

```
<?xml version="1.0" encoding="utf-8"?>
<satyam><!-- This is a root element -->
  <eltptraining phase="1" days="22"><!-- phase and days are attribute here -->
    <course>
      <module duration="7 days">C and DS</module>
      <module duration="2 days">Operating System </module>
      <module duration="3 days">RDBMS </module>
```

```
<module duration ="2 days">Web Programming </module>
<module duration ="8 days">Software Engineering</module>
</course>
</eltpttraining>
</satyam><!--End of root element-->
```



Practice Session :

- Write an XML document providing personal details like name, address, age, sex and phone no using proper elements and appropriate attributes along with comments.



Check List :

- Understanding of elements and attributes of XML document.
- Use of proper naming conventions for elements and attributes.



Common Errors :

- Improper names used for elements and attributes.
- Improper nesting of elements.



Lessons Learnt:

- ☒ How to write XML documents.



Best Practices:

- Make it a practice to write XML document along with comments.

**Topic: XML Namespaces****Estimated Time: 90 Mins.****Objectives :** This module will familiarize the participant with

- XML Namespaces and usage of XML Namespace to avoid name conflicts.

**Presentation :**

There are two methods to avoid naming conflicts in an XML document. They are

Name conflicts resolved by prefix.

Naming conflicts occur when two xml files use the same attribute name. To avoid these naming conflicts prefix is used for the attributes, so that both can be differentiated by XML parser.

With naming conflict	Without naming conflict
<table> tag from html tag	<h:table>
<table> user defined tag for <furniture> element	<f:table> using prefix

Name conflicts resolved by xmlns attribute.

Xmlns attribute is used to differentiate between 2 attributes, so that the naming conflicts can be resolved.

With naming conflict	Without naming conflict
<table> tag from html tag	<h:table xmlns:h="http://www.abc.com/furniture/">
<table> user defined tag for furniture element	<f:table xmlns:f="http://www.abc.com/furniture/">

**Scenario:**

Vaibhav Furniture is a furniture shop. It maintains a record of the customers and furniture sold in XML files. Suppose one XML file has HTML tag <table> and another <table> tag for element Furniture. To enter customer name, html table and <table> tags are used. In this case XML parser will not be able to differentiate between <table> tags and <table> attributes. Hence naming conflicts occur. The solution to this is provided below.



Demonstration/Code Snippet :

Name conflicts solved by prefix:

Step 1: Create an XML file in visual studio 2005, file menu → new file → XML file → "furniture.xml", which contains HTML <table> attribute and <table> for attribute furniture..Here extension of the xml file is .xml.

```
<Personal_furniture>
<table>
  <tr>
    <td>Bhrahmagna</td>
    <td>Trivedi</td>
  </tr>
</table>

<table>
  <name>Dining Table</name>
  <width>80</width>
  <Height>30</Height>
</table>
</Personal_furniture>
```

Step 2: So both the <table> attribute cannot be differentiated by xml parser. To avoid naming conflicts prefix is used.

```
<Personal_furniture >
<h:table>
  <h:tr>
    <h:td>Bhrahmagna</h:td>
    <h:td>Trivedi</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>Square Table</f:name>
  <f:width>80</f:width>
  <f:Height>80</f:Height>
</f:table>
</Personal_furniture>
```

Name conflicts solved by xmlns attribute:

- When we are using xmlns attribute we have to give prefix to the xml element.
- This namespace is defined by xmlns attribute as the first attribute of an element
- Syntax for that is, xmlns: prefix="URI".



URI: A **Uniform Resource Identifier** (URI) is a string of characters which uniquely identifies an Internet Resource. The most common URI is the **Uniform Resource Locator** (URL) which identifies an Internet domain address. Another, not so common type of URI is the **Universal Resource Name** (URN). In our examples we will only use URLs.

Step 1: Create an XML file in visual studio 2005, file menu → new file → XML file → "furniwithxmlns.xml", which contains HTML <table> attribute and <table> for attribute furniture.. Here extension of the xml file is **.xml**.

```
< Personal_furniture>
<h:table xmlns:h="http://www.abc.com/furniture/">
    <h:tr>
        <h:td>Bhrahmagna</h:td>
        <h:td>Trivedi</h:td>
    </h:tr>
</h:table>

<f:table xmlns:f="http://www.abc.com/furniture/">
    <f:name>Square Table</f:name>
    <f:width>80</f:width>
    <f:Height>80</f:Height>
</f:table>
</Personal_furniture>
```

We can also define both xmlns attribute in root element only:

```
<Personal_furniture xmlns:h="http://www.abc.com/html/"  
xmlns:f="http://www.abc.com/furniture/">  
.  
.  
.  
</Personal_furniture >
```



Context :

- XML naming conflicts is avoided by prefix and XML namespaces attribute xmlns.



Practice Session :

- Create an XML document showing student details. Create another XML document showing details of the students appearing for online examination. After creating both the documents try to solve the naming conflicts using prefix method.



Check List :

- XML attribute naming Conflicts.
- Two ways to avoid naming conflicts.



Common Errors :

- Syntax not properly defined.



Lessons Learnt :

- ☒ XML document structure



Best Practices :

- Try to use namespace attributes initially.



Avoid name conflicts: use xml namespace attributes initially.



Topic: **XML Tree**

Estimated Time: 60 Mins.



Objectives : This module will familiarize the participant with

- Building and Understanding XML Tree structure



Presentation :

- XML File represented as a Tree Structure.
- XML documents must contain a **root element**. This element is "the parent" of all other elements.
- The elements in an XML document form a document tree. The tree starts from the root and branches to the lowest level of the tree.
- All elements can have sub elements (child elements).
- Syntax of XML document tree is as follows:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```



Scenario :

- Osmania University provides a wide variety of courses to its students. It maintains a library with a large variety of books. The details of the books can be shown in XML files in a tree structure. Here is a sample document which shows the details of 3 different books. A library file "library.xml" describes the structure of the library. Root element is <library> and child element is <book> which lists different books in the library. <book> element has an attribute called *category* that describes different categories like .net, java and web. <book> has child elements <title> - title of the book, <author> - author of the book, <year> - year of book published and <price> price of the book. <title> has an attribute "lang", that defines the language of the book.



Demonstration/Code Snippet:

XML Document - > library.xml:

```

<Library>
  <book category=".NET">
    <title lang="en"> Learn VB.Net</title>
    <author>Paul James</author>
    <year>2005</year>
    <price>300.00</price>
  </book>
  <book category="Java">
    <title lang="en">J2EE</title>
    <author>E.Balaguruswamy</author>
    <year>2007</year>
    <price>450</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>390</price>
  </book>
</Library>

```

XML Document Tree Structure:

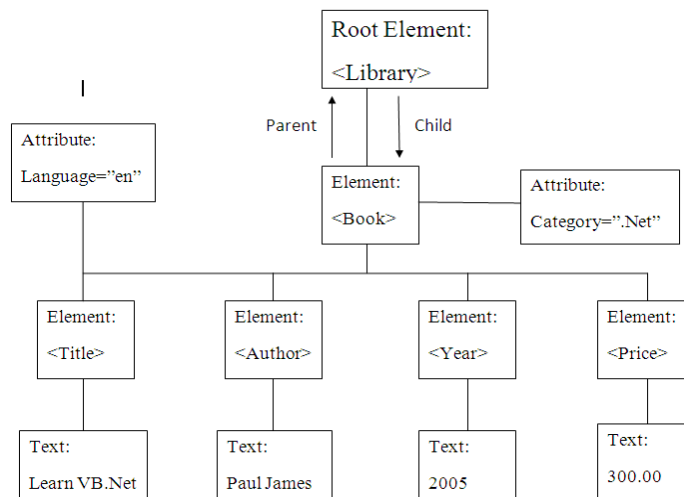


Figure 1.4-1: XML Document Tree Structure



Context :

- XML document is a tree structure.



Practice Session :

- Create a tree structure for the XML document with student details.



Check List :

- XML Tree structure.



Common Errors :

- Syntax not properly defined.
- Closing tag of an element not defined



Lessons Learnt :

- ☒ Basics of XML Document.
- ☒ XML Document as a tree Structure.



Best Practices :

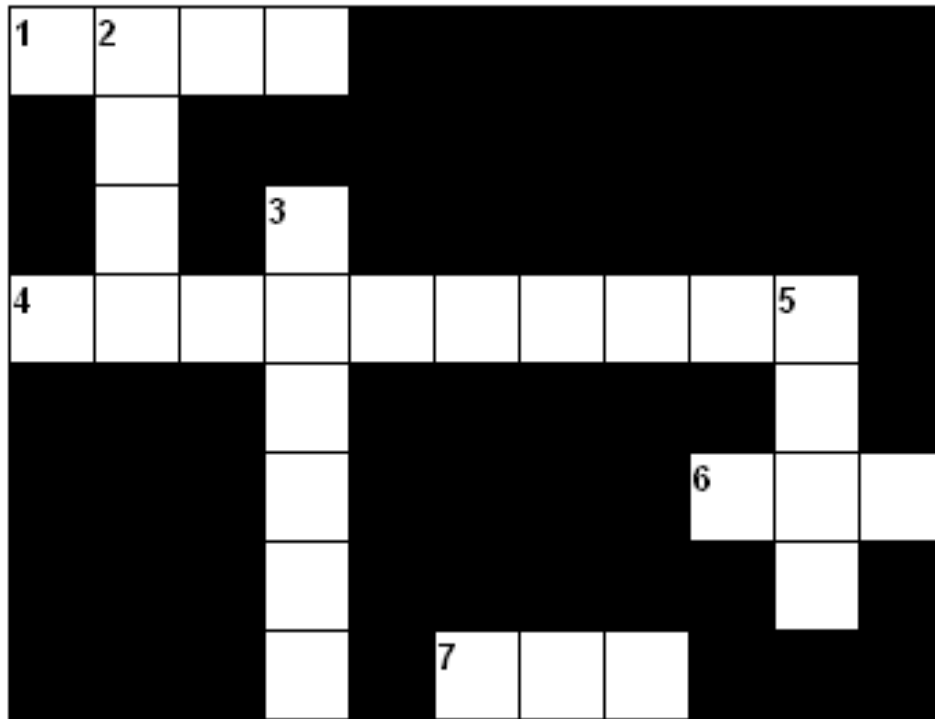
- Follow the tree structure while creating XML documents



Avoid error: Best practice is to complete the first element with closing tag and then use child elements inside first element.

CROSSWORD UNIT 1

Time: 20 minutes



Across:

1. XML File is a ____ structure.
4. ____ contain values that are associated with an element.
6. XML file cannot start with non-alphabetical characters or numbers or the letters ____.
7. XML is recommended by _____.

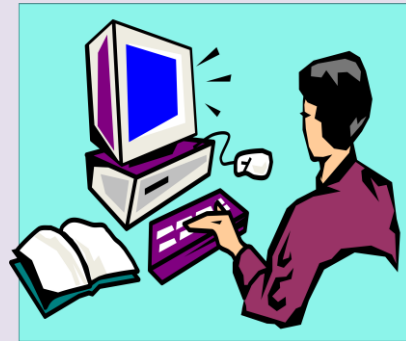
Down:

2. XML documents must contain ____ element.
3. To avoid naming conflicts ____ is used for attributes, so both can be differentiated by XML parser.
5. XML is interoperable with _____.

2.0 XML Document

Topics

- ✚ 2.1 XML Document Structure and Syntax
- ✚ 2.2 DTD (Document Type Definition)
- ✚ 2.3 XSD (XML Schema Definition)



Topic: **XML document structure and syntax**

Estimated Time: 75 mins.



Objectives : This module will familiarize the participant with

- Structure of XML document.
- Syntax used in XML document.



Presentation :

- XML document structure is divided into different parts. It starts with XML declaration, parent element (root), child element and text of all elements.
- Different syntax used in XML document is empty elements, entity reference and special characters.



Scenario :

- Osmania University provides a wide variety of courses to its students. It maintains a library with a large variety of books. The details of the books can be shown in XML files in a tree structure. Here is a sample document which shows the details of 3 different books. A library file “library.xml”, describes the structure of the library. Root element is <library> and child element is <book>, that lists different books. The <book> element has attribute called *category* which describes different categories like .net, java and web. The <book> element has child elements, <title> - title of the book, < author> - author of the book, < year> - year of book published and <price> price of the book. The <title> element has an attribute “lang”, which defines the language of the book.



Demonstration/Code Snippet :

Step 1: Create a new XML document of library which contains book, title, author elements and main root elements library.

```
<?xml version="1.0" encoding="utf-8" ?>
<Library>
  <book category=".NET">
    <title lang="en"> Learn VB.Net</title>
    <author>Paul James</author>
    <year>2005</year>
    <price>300.00</price>
    <availability/>
  </book>
  <book category="Java">
    <title lang="en">J2EE</title>
```

```
<author>J K. Rowling</author>
    <year>2007</year>
    <price>450</price>
    <availability/>
</book>
<book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>390</price>
    <availability/>
</book>
</Library>
```



Explanation for the above XML file

Here, `<?xml version="1.0" encoding="utf-8" ?>` is a declaration tag which shows one of the most common formats of an XML declaration, with an XML version of 1.0 and an encoding style of utf-8. For other languages encoding style can be different.

`<Library>` is root element which is closed by end tag `</Library>`. `<book>` is first element enclosed by end tag `</book>` which has attribute category. `<title>`, `<author>`, `<year>`, `<price>` and `<availability/>` are also child elements of the element `<book>`.

`<availability/>` is empty element, which is often kept in XML output to accommodate a predefined data structure that is specified for data validation in DTD – Document Type Definition or Schema.



Elements and Attribute in an XML

In a XML document, Library is root element and book is first element which has attribute category. . This category can also be written as element. In the similar manner Lang can also be written as element



Context :

Use Different syntax and attributes in xml document as well as for an empty element.



Practice Session :

Create a XML document for student data. Use different syntax and also use one empty element this would be useful for validation of DTD.



Check List :

- XML document structure and syntax.



Common Errors :

- Syntax not properly defined.
- End tag of element seems to be incomplete
- Empty element need not have an end tag.



Lessons Learnt :

- ☒ XML document structure



Best Practices :

- Always use empty elements for validation of DTD.



Wasting memory: Use empty elements only when needed else it is wastage of memory.



Topic: DTD (Document Type Definition)

Estimated Time: 120 mins.



Objectives : This module will familiarize the participant with

- Introduction to DTD.
- Internal & External declaration of DTD.
- Use of DTD.
- XML document structure from a DTD point of view.



Presentation :

Introduction to DTD:

- A Document Type Definition (DTD) defines the legal building blocks of an XML document. It describes the document structure with a list of legal elements and attributes.
- A DTD can be declared inside an XML document or within a separate external DTD file.
- A DTD declared inside a document is known as internal DTD whereas if declared outside the XML document it is known as external DTD.

Internal & External Declaration of DTD:

Declaration of DTD can be done in two ways, Internal & External.

Use of DTD:

- With a DTD, every xml file carries description of its own format. With the help of DTD, different group of user can share standard DTD for exchanging data. With the help of a DTD we can validate the data we receive from the outside world as well as our own data.

XML Document structure from a DTD point of view:

From a DTD point of view, XML document is made of following building blocks.

- Elements
- Attributes
- Entities
- PCDATA
- CDATA

Elements:

- Elements are the most basic building blocks of an XML and HTML document.
- It describes the XML document's elements and optional nested elements.
- `<form>` and `<head>` are the example elements for an html document while `<ADDRESS>` and `<NAME>` are the examples of elements for above described xml files.

Please note that for an html file the elements are pre-defined while for xml files the elements are author defined.

Attributes:

- This declaration describes attributes and optional values within an XML Document.
- The attribute type can be CDATA, ID, IDREF, NMTOKEN etc.

Entities:

- Defined shortcuts for standard text or special characters.
- Entity examples are as below

< <

> >

& &

- Entities are expanded when XML document is parsed.

PCDATA (Parsed Character Data)

- It is the text found between start tag and end tag of an XML element.
- It is the text parsed and examined by parser.
- PCDATA should not contain symbols such as <, > or &. They should be represented using <, > or &.

CDATA (Character Data)

- It is the text that will not be parsed by parser.

**Scenario :**

Ram is a student of Osmania University. He has shifted to a new home. He wants to update his new address in the University records. He prepares an XML document that shows the address, divided into different parts like Name, Careof, ResidenceName, StreetName, City, Pincode and State.

Use DTD to show all parts in a single address, in both the ways, internal and external.

**Demonstration/Code Snippet :****Internal Declaration:**

Step 1: Create a new XML document address.xml, which shows name, residence name, street name, city, pin code, state.

Address.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ADDRESS [
<!ELEMENT ADDRESS
  (NAME,CAREOF,RESIDENCENAME,STREETNAME,CITY,PINCODE,STATE)>
  <!ELEMENT NAME (#PCDATA)>
  <!ELEMENT CAREOF (#PCDATA)>
  <!ELEMENT RESIDENCENAME (#PCDATA)>
  <!ELEMENT STREETNAME (#PCDATA)>
  <!ELEMENT CITY (#PCDATA)>
  <!ELEMENT PINCODE (#PCDATA)>
  <!ELEMENT STATE (#PCDATA)>
]>
<ADDRESS>
  <NAME>MR. A.B.KHANNA</NAME>
  <CAREOF>MR. B.H.KHANNA</CAREOF>
  <RESIDENCENAME>21,Vihar Appt.</RESIDENCENAME>
  <STREETNAME>Chorangi Lane,maninagar</STREETNAME>
  <CITY>Ahmedabad</CITY>
  <PINCODE>380008</PINCODE>
  <STATE>Gujarat</STATE>
</ADDRESS>
```

External Declaration:

External.dtd



Use of External DTD: This file declares the dtd structure for address.

```
<!ELEMENT ADDRESS
  (NAME,CAREOF,RESIDENCENAME,STREETNAME,CITY,PINCODE,STATE)>
  <!ELEMENT NAME (#PCDATA)>
  <!ELEMENT CAREOF (#PCDATA)>
  <!ELEMENT RESIDENCENAME (#PCDATA)>
  <!ELEMENT STREETNAME (#PCDATA)>
  <!ELEMENT CITY (#PCDATA)>
  <!ELEMENT PINCODE (#PCDATA)>
  <!ELEMENT STATE (#PCDATA)>
```

Externaldtd.xml



Use of External DTD: External DTD file is used in XML file in document declaration.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ADDRESS SYSTEM "address.dtd">
<ADDRESS>
  <NAME>MR. A.B.KHANNA</NAME>
  <CAREOF>MR. B.H.KHANNA</CAREOF>
  <RESIDENCENAME>21,Vihar Appt.</RESIDENCENAME>
  <STREETNAME>Chorangi Lane,maninagar</STREETNAME>
  <CITY>Ahmedabad</CITY>
  <PINCODE>380008</PINCODE>
  <STATE>Gujarat</STATE>
</ADDRESS>
```



Context :

- Introduction to DTD.
- Internal & External DTD.
- Use of DTD.



Practice Session :

- Create a DTD for student data and use it in XML file in both external and internal way. Student Data is divided into different parts like SID, Name, Age, Stream, Address, Phone and Percentage.



Check List :

- DTD & Use of DTD.



Common Errors :

- Syntax not properly defined.
- In External DTD file syntax Error. Sentence can improved technically



Lessons Learnt :

- ☒ DTD and its use.
- ☒ Internal & External declaration of DTD.
- ☒ XML structure from a DTD point of view.



Best Practices :

- Use External DTD file because it can be used by another application or XML file.



Topic: XSD (XML schema definition)

Estimated Time: 120 mins.



Objectives : This module will familiarize the participant with

- Introduction to XSD.
- Advantages of XSD.



Presentation :

- XML schema definition (XSD), recommendation of W3C, specifies how to formally describe the elements in an XML document. This description can be used to verify that each item of content in a document adheres to the description of the element in XSD.
- In general, a schema is an abstract representation of an object's characteristics and relationship to other objects. An XML schema represents the interrelationship between the attributes and elements of an XML object (for example, a document or a portion of a document). To create a schema for a document, first analyze its structure, and then define each structural element as it is encountered. Document structure is divided into different parts. It starts with XML declaration, parent element (root), child element and text of all elements.

- XSD has several **advantages** over earlier XML schema languages, such as document type definition (DTD) or Simple Object XML (SOX). For example, it's more direct: XSD, in contrast to the earlier languages, is written in XML, which means that it doesn't require intermediary processing by a parser. Other benefits include self-documentation, automatic schema creation, and the ability to be queried through XML Transformations (XSLT). Despite the advantages of XSD, it has some detractors who claim, for example, that the language is unnecessarily complex.



Scenario :

to ensure a reliable database of the books in the library, Osmania University has a detailed description of the books in the database maintained in XML by the use of XSD along with XML files.



Demonstration/Code Snippet :

Step 1: Create a new XML Schema file, library.xsd, which contains book, title, author elements and main root elements library.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xsd:element name="bookstore" type="bookstoreType"/>
  <xsd:complexType name="bookstoreType">
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="book" type="bookType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="bookType">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" type="authorName"/>
      <xsd:element name="price" type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="genre" type="xsd:string"/>
    <xsd:attribute name="publicationdate" type="xsd:date"/>
    <xsd:attribute name="ISBN" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="authorName">
    <xsd:sequence>
      <xsd:element name="first-name" type="xsd:string"/>
      <xsd:element name="last-name" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```



Structure of XSD element: XSD is prefix in every element, type of element means sequence, complex or else. Then name and type of element and

attributes in other child tags.

Step 2: Structure of library.xsd file is shown below:

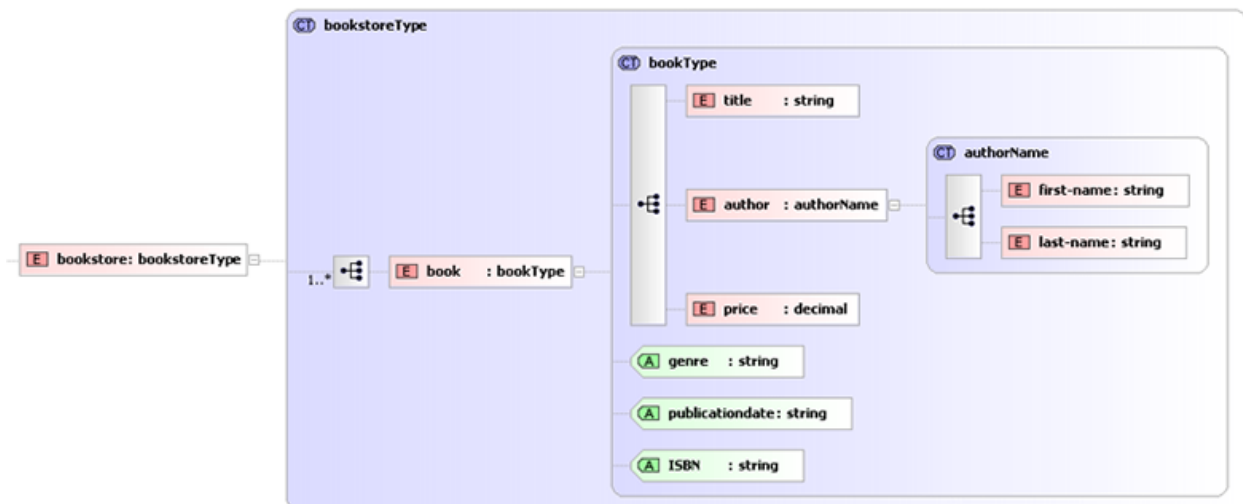


Figure 2.3-1: Structure of library.xsd file

Step 3: Sample example XML file is used for above example.

```
<bookstore>
  <book genre="autobiography" publicationdate="1981" ISBN="1-
861003-11-0">
    <title>The Autobiography of Benjamin Franklin</title>
    <author>
      <first-name>Benjamin</first-name>
      <last-name>Franklin</last-name>
    </author>
    <price>8.99</price>
  </book>
</bookstore>
```

Now, we will try to create XML Schema using Visual Studio 2005.

Step 1: Go to Add New Item

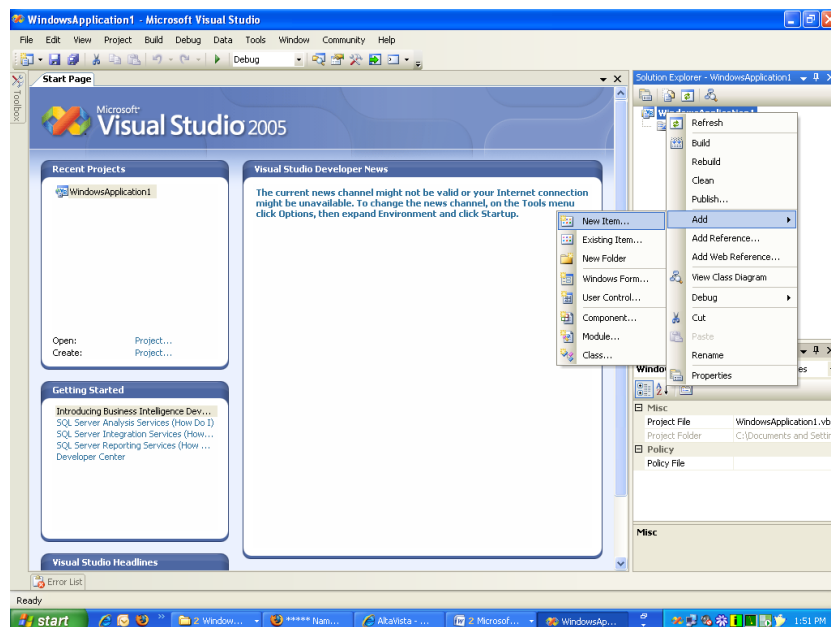


Figure 2.3-2: Adding new item

Step 2: Select XML Schema File

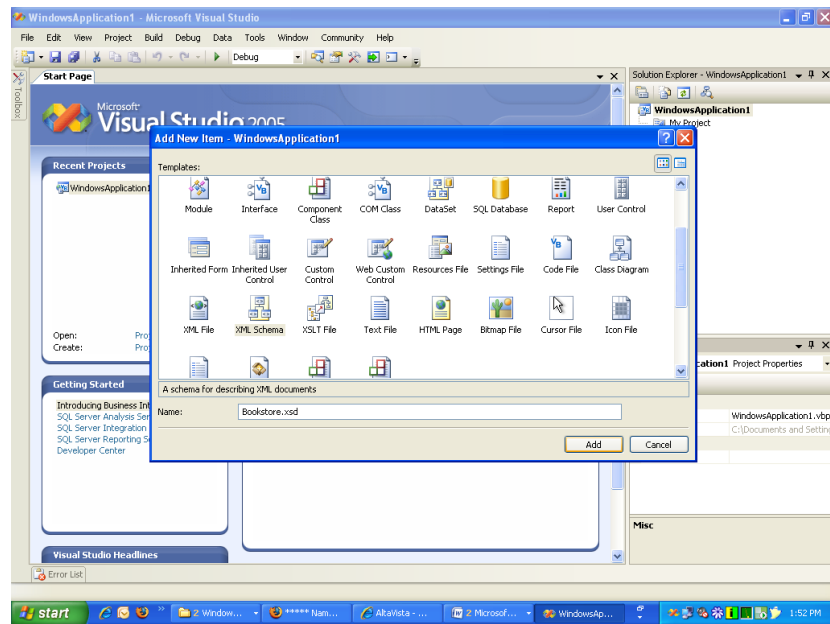


Figure 2.3-3: Selecting XML Schema file

Step 3: You will get a tool bar through which you can add different blocks.

Example: element, complex type etc.

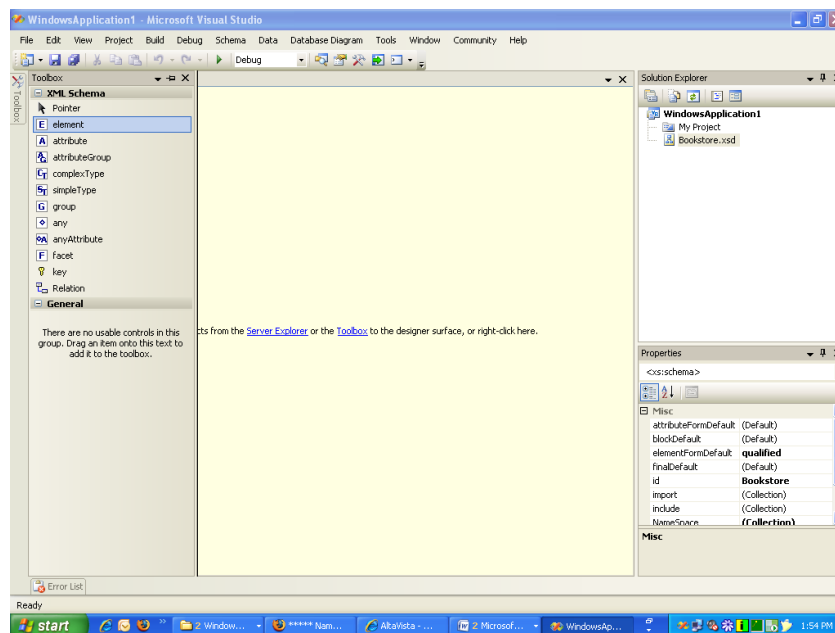


Figure 2.3-4: Adding different blocks

Step 4: Select an element

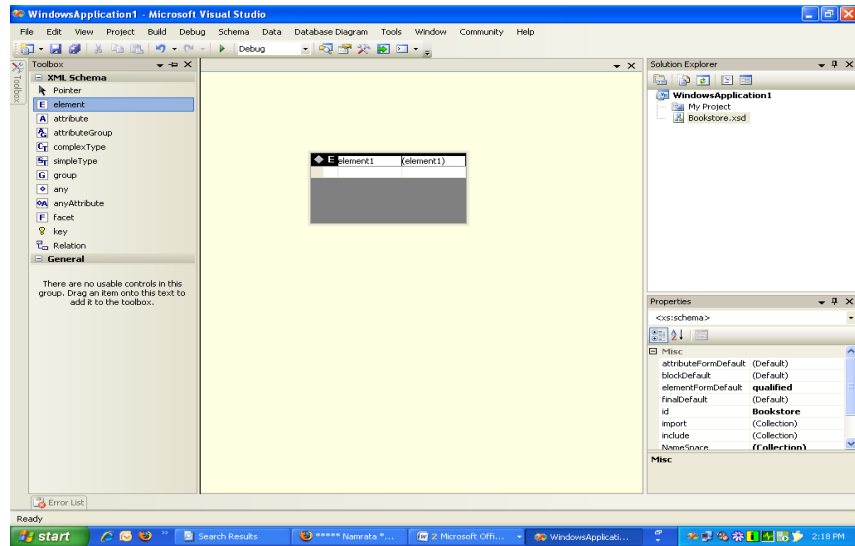


Figure 2.3-5: Selecting elements

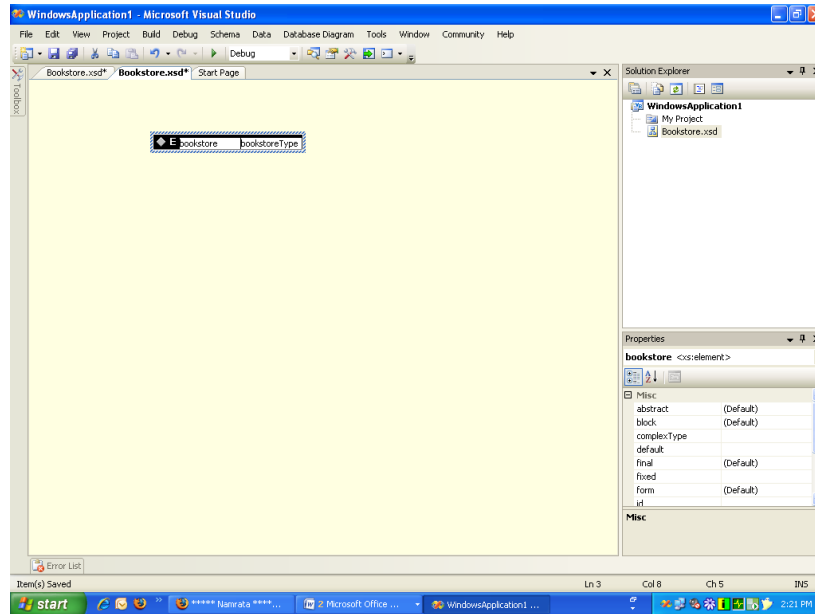


Figure 2.3-6: Selecting elements

Step 5: Add a Complex Type

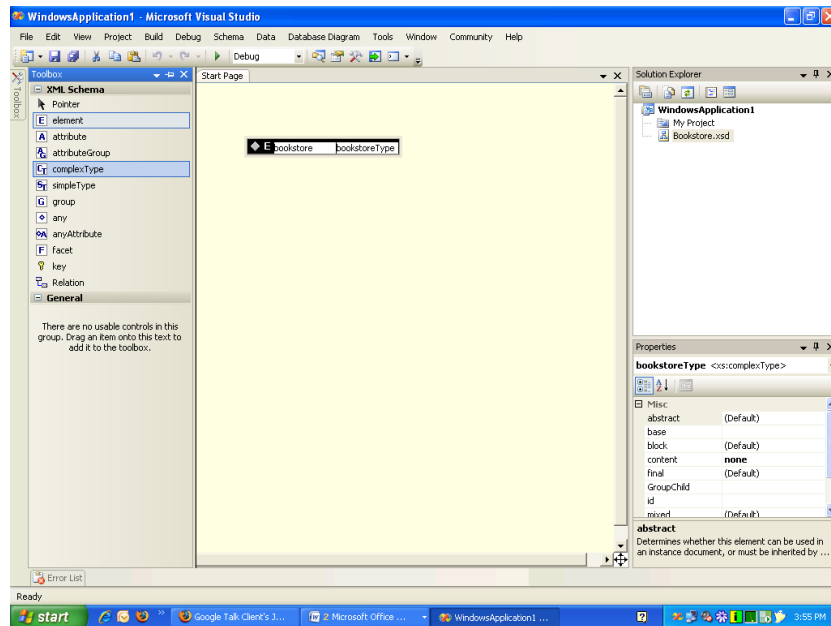


Figure 2.3-7:

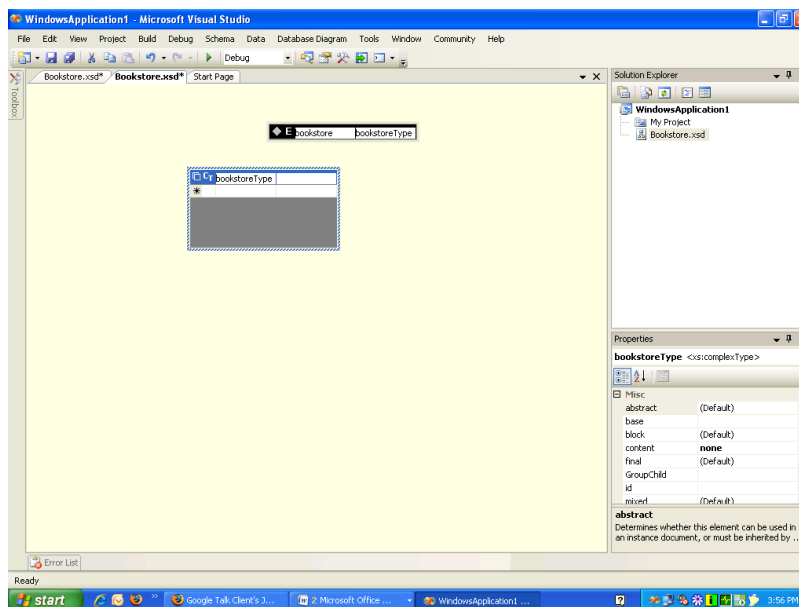


Figure 2.3-8:

Step 6: Add a Sequence

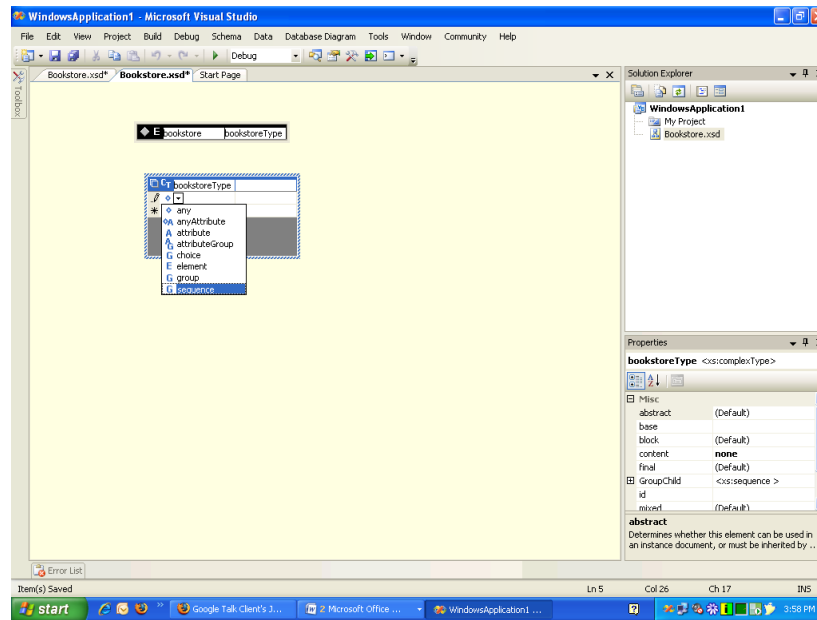


Figure 2.3-9:

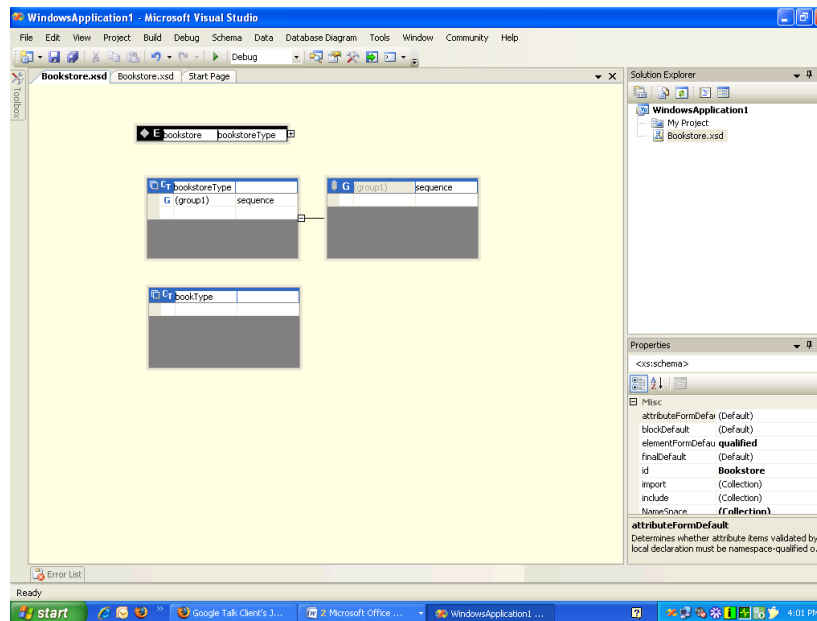


Figure 2.3-10:

Step 7: Thus by adding different building blocks we can create our own XML Schema with visual approach.

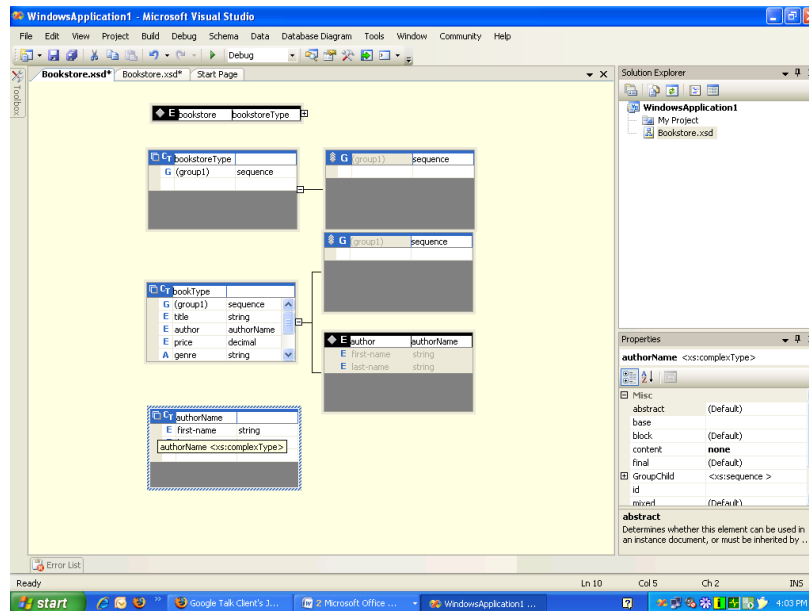


Figure 2.3-11:



Context :

- Use of XSD over DTD



Practice Session :

- Create an XML schema (XSD file) for student data and use it in XML file.



Check List :

- XSD and Advantages of XSD.



Common Errors :

- Syntax not properly defined.
- Element starts with XSD prefix then name followed by type of element.



Lessons Learnt :

- ☒ XSD and its advantages.

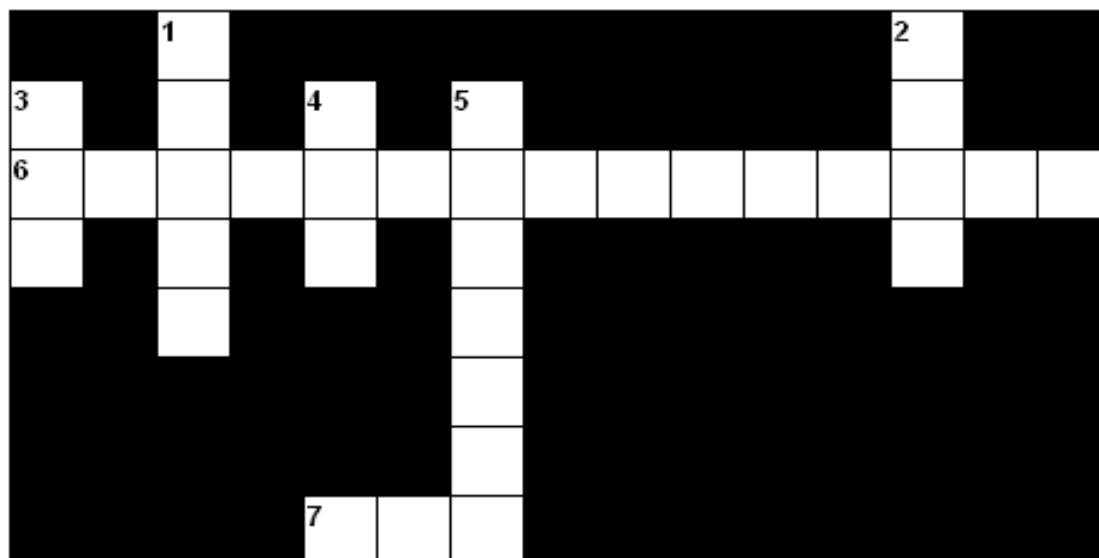


Best Practices :

- XSD is advantageous than earlier XML schema file, hence use XSD schema.

CROSSWORD UNIT 2

Time: 20 minutes



Across:

6. XML can be queried through XML _____.
7. Earlier XML schema languages before XML are document type

Down:

1. From a DTD point of view, XML document is made of following building blocks Elements, Attributes, Entities, PCDATA and _____.
2. XML document structure is divided into different parts,



definition (DTD) or _____.

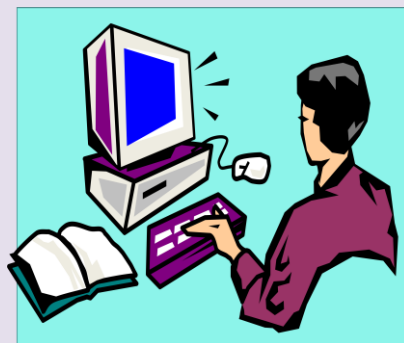
starts with XML declaration, parent element also called as _____, child element and text of all elements.

3. Always use empty elements for validation of _____.
4. _____ recommendation of W3C, specifies how to formally describe the elements in an XML document
5. Type of element are sequence, _____ etc.

3.0 XML Parsing Concepts

Topics

-  3.1 XML DOM (Document Object Model)
-  3.2 SAX and other XML Parsers



**Topic: XML DOM (Document Object Model)****Estimated Time: 120 mins.****Objectives :** This module will familiarize the participant with

- DOM and its need for processing documents on the client side
- Parsing an XML document with DOM
- Handling errors in XML DOM

**Presentation :****What is DOM?**

- DOM is a memory model for representing elements by W3C standard for accessing, navigating and manipulating XML documents.
- Its API allows programs and scripts to dynamically access and update the content, structure and style of a document.
- It is a tree structure representation with elements, attributes and text as nodes and is called as node tree.
- The root of the tree is a document node which has one or more child nodes.
- Every XML element is an element node, every attribute is an attribute node and the text in the XML elements are text nodes.

XML document example:

```
<?xml version="1.0" encoding="utf-8" ?>
<satyam>
  <eltp_training course="1">
    <course_name>VB.Net </course_name>
    <course_id>vb1</course_id>
    <course_duration>30 days </course_duration>
  </eltp_training>

  <eltp_training course="2">
    <course_name>Java </course_name>
    <course_id>jv1</course_id>
    <course_duration>30 days </course_duration>
  </eltp_training>
</satyam>
```

When the above XML file (satyam.xml) is loaded into DOM, DOM loads the XML file into a tree like structure.

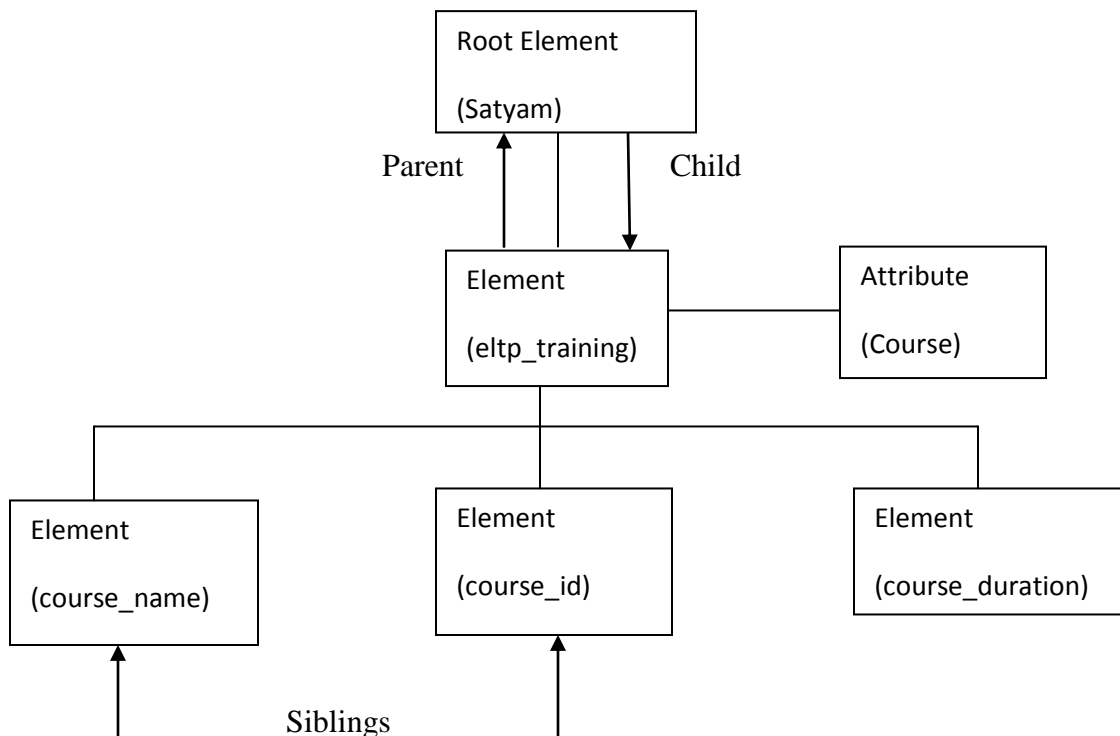


Figure 3.1-1: Tree Structure of XML Document Example

Why client-side XML DOM processing?

- It reduces server load and gives the visitor a better, more responsive experience on the site.

Parsing XML:

- All modern browsers have a built-in XML parser that can be used to read and manipulate XML.
- The parser reads XML into memory and converts it into an XML DOM object that can be accessed through JavaScript.
- There are some differences between Microsoft's XML parser and the parsers used in other browsers. The Microsoft parser supports loading of both XML files and XML strings (text), while other browsers use separate parsers. However, all parsers contain functions to traverse XML trees, access, insert, and delete nodes.

DOM Levels:

According to W3C recommendations,

- DOM Level 1 allows navigation within an HTML or XML document and the manipulation of its content.
- DOM Level 2 extends Level 1 with extra features such as XML namespace support, filtered views, ranges and events.
- DOM Level 3 builds on Level 2 that allows programs to dynamically access and update the contents, structure and style of documents.

There are many interfaces defined that form the DOM Level 3 core module.

Document interface is the uppermost object in the XML DOM hierarchy.

- It provides methods for navigating, querying and modifying the contents and structure of an XML document.
- The important methods are createElement, createAttribute, createTextNode, CreateNode, CreateComment, getElementsByTagName, load, loadXML, transformNode, save.

Loading an XML Document (satyam.xml)

To traverse an XML Document in Internet Explorer, first you need to instantiate the Microsoft XMLDOM parser. In Internet Explorer 5.0 and above, the parser can be instantiated using JavaScript:

To write the javascript go to Visual Studio 2005->File->New Website->default.aspx file->source view, add <script> tag in <head> tag write the code below:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
    <script type="text/javascript" language="javascript">
function loadDocument ()
{
var xmlDoc=new ActiveXObject ("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.load ("satyam.xml");
alert("File Loaded");
}
</script>

</head>
<body>
    <form id="form1" runat="server">
        <div>
        </div>
    </form>
</body>
</html>
```

- In the function loadDocument, the first line creates an empty Microsoft XML document object.
- The second line turns off asynchronized loading, to make sure that the parser will not continue execution of the script before the document is fully loaded.
- The third line tells the parser to load an XML document called "satyam.xml".

The following JavaScript code loads a string called *txt* into the parser.

```
<head runat="server">
    <title>Untitled Page</title>
    <script type="text/javascript" language="javascript">
function loadText ()
{
var xmlDoc=new ActiveXObject ("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.loadXML(txt);
alert("String loaded")
}

</script>
</head>
```

Note: The **loadXML()** method is used for loading strings (text), **load()** is used for loading files.

XML Parser in Firefox and Other Browsers

The following JavaScript fragment loads an XML document ("[satyam.xml](#)") into the parser:

```
xmlDoc=document.implementation.createDocument("", "", null);  
xmlDoc.async="false";  
xmlDoc.load("satyam.xml");
```

- The first line creates an empty XML document object.
- The second line turns off asynchronous loading, to make sure that the parser will not continue execution of the script before the document is fully loaded.
- The third line tells the parser to load an XML document called "satyam.xml".

The following JavaScript fragment loads a string called txt into the parser:

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

- The first line creates an empty XML document object.
- The second line tells the parser to load a string called txt.

Note: Internet Explorer uses the **loadXML()** method to parse an XML string, while other browsers uses the **DOMParser** object.

Handling Errors in XML DOM:

- The `parseError` property of the Document object gives us information about the exceptions and errors.
- It is derived from the interface `IXMLDOMParseError`.
- The properties of the `IXMLDOMParseError` object are `reason`, `line`, `errorCode`, `linepos` and `srcText`.

EXAMPLE:

```
<head runat="server">
  <title>Untitled Page</title>
  <script type="text/javascript" language="javascript">
    function errhand()
    {
      var doc=new ActiveXObject("Microsoft.XMLDOM");
      doc.load("satyam.xml");
      if (doc.parseError.errorCode != 0)
      {
        alert ("Error Code: " + doc.parseError.errorCode);
        alert ("Error Reason: " + doc.parseError.reason);
        alert ("Error Line: " + doc.parseError.line);
      }
      else
      {
        alert (doc.documentElement.xml);
      }
    }
  </script>
</head>
```

In this code, first a DOM object is created and then the *if* construct is used to determine whether the *parseError* property of this object returns any error code. If error code is there, then the details of the error indicating the error code, reason and the line number where the error has occurred is displayed or else only a message box is displayed showing the XML of the document.



Scenario :

Satyam.xml is an XML document which describes two of the training courses at Satyam. It is required to load this file into the browser. The code given below demonstrates to load an XML document in the browser using functions in DOM.



Demonstration/Code Snippet :

- The XML DOM contains methods (functions) to traverse XML trees, access, insert, and delete nodes.
- However, before an XML document can be accessed and manipulated, it must be loaded into an XML DOM object.
- Earlier it was demonstrated how to load XML documents. To make it simpler, it should be written as a function:


```
<head runat="server">
  <title>Untitled Page</title>
  <script type="text/javascript" language="javascript">
function loadXMLDoc (dname)
{
  try //Internet Explorer
  {
    xmlDoc=new ActiveXObject ("Microsoft.XMLDOM");
  }
  catch (e)
  {
    try //Firefox, Mozilla, Opera, etc.
    {
      xmlDoc=document.implementation.createDocument ("", "", null);
    }
    catch (e) {alert (e.message) }
  }
  try
  {
    xmlDoc.async=false;
    xmlDoc.load (dname);
    return (xmlDoc);
  }
  catch (e) {alert (e.message) }
  return (null);
}
</script>
</head>
```

To make the function above even easier to maintain, and to make sure the same code is used in all pages, it can be stored in an external file.

The file "loadxmldoc.js" is called.

The file can be loaded in the <head> section of an HTML page, and loadXMLDoc() can be called from a script in the page:

```
<html>
<head>
<script type="text/javascript" src="loadxmldoc.js">
</script>
</head>

<body>
<script type="text/javascript">
xmlDoc=loadXMLDoc("satyam.xml");
document.write("xmlDoc is loaded, ready for use");
</script>
</body>
</html>
```



Context :

- To understand the tree structure representation of DOM.
- Parsing XML with DOM



Practice Session :

- Write an XML document describing your course details and understand the tree structure representation of that.
- Parse that document with DOM



Check List :

- XML DOM object model
- Parsing XML with DOM
- Handling Errors in XML DOM



Common Errors :

- Expecting an element node to contain text.



Exceptions :

- For security reasons, modern browsers do not allow access across application domains.
- This means both the web page and the XML file must be located on the same server. Otherwise the xmlDoc.load () method, will generate the error "Access is denied".



Lessons Learnt :

- ☒ Tree structure representation of DOM
- ☒ Loading an XML Document in the browser



Best Practices :



Use of DOM: For DOTNET framework use dom instead of SAX(Simple API for XML).



Topic: **SAX and other XML parsers**

Estimated time: 90 mins.



Objectives : This module will familiarize the participant with

- Introduction SAX
- Concepts of different XML parsers.
- Comparison between DOM and SAX.



Presentation:

Introduction to SAX

- SAX implies Simple API for XML.
- Like DOM, SAX is also used to describe, parse and manipulate XML documents.
- Unlike DOM, SAX breaks a document down into a series of events that represents parts of an XML document, such as StartDocument, StartElement, EndElement, ProcessingInstruction, SAXError and EndDocument.
- Unlike DOM, SAX is an event-driven API. Rather than building an in-memory copy of the document and passing it to the client program, it requires the client program to register itself to receive notifications when the parser receives various parts of an XML document.
- SAX is not developed or recommended by the W3C.
- SAX parsing is faster than DOM parsing, but slightly more complicated to code.
- SAX is not an XML parser; instead it is a set of interfaces implemented by many XML parsers.
- SAX 1 supports navigation around a document and manipulation of content via SAX 1 events via the SAX 1 parser class.
- SAX 2 supports namespaces, filter chains, and querying and setting features and properties via SAX events via the SAX 2 XMLReader interface.

About XML Parsers:

The most accepted parsers among all the parsers are Apache Xerces, IBM XML4J (XML for Java) and Microsoft's MSXML parser.

Parsers are generally divided into 2 categories:

Non-validating parsers check that an XML document adheres to basic XML structure and syntax rules (well-formed XML)

Validating parsers verify that an XML document is valid according to the rules of a DTD or schema, as well as checking for a well-formed document structure and syntax.

The latest versions of the parsers discussed above are validating parsers.

Apache's Xerces

- This is a validating parser that is available in Java and C++.
- It fully supports the W3C XML DOM (Levels 1, 2, 3) standards, and SAX version 2.
- It is a validating parser and provides support for XML document validation against W3C schemas and DTDs.

IBM's XML4J

- The IBM XML for Java (XML4J) libraries are based on Xerces and support W3C XMLSchema Recommendation when implementing the validating parser interfaces.
- It supports SAX 1 and 2, DOM 1 and 2 and some basic features of DOM 3 standard, currently in the recommendation process.

Microsoft's XML Parser (MSXML)

- It is a part of Internet Explorer 5.5 or later, and the latest version is separated from the IE browser code, so that the parser does not have to wait for the next version of the browser, and vice versa.
- It supports most XML standards and works with JavaScript (and DHTML), Visual Basic, ASP, and C++, but not Java.
- It includes support for DOM, XML Schema definition language for validating parsers, Schema Object Model (SOM, a Microsoft invention which parses XML schemas into an Object Model), XSLT, XPath and SAX.

Comparison between DOM and SAX:

- SAX is better at parsing larger documents. If the application uses smaller documents or needs to navigate an XML document more than once, DOM parsing is more suitable.
- SAX is very good at parsing parts of a large document efficiently, but SAX parses through a document once to collect needed data, it has to start over as more document data is needed. On the other hand, DOM holds a node tree in memory until the application is finished with it, so once a document is parsed, pieces of the document can be retrieved without having to re-parse the document.



Context :

- To understand SAX and about all other parsers.
- Comparison between DOM and SAX parser.



Practice Session :

Parse the Satyam.xml file using different parsers.



Check List :

- Concept of SAX.
- Concept of other XML parsers.

**Common Errors :**

- Parsing a Java file using MSXML parser.

**Lessons Learnt :**

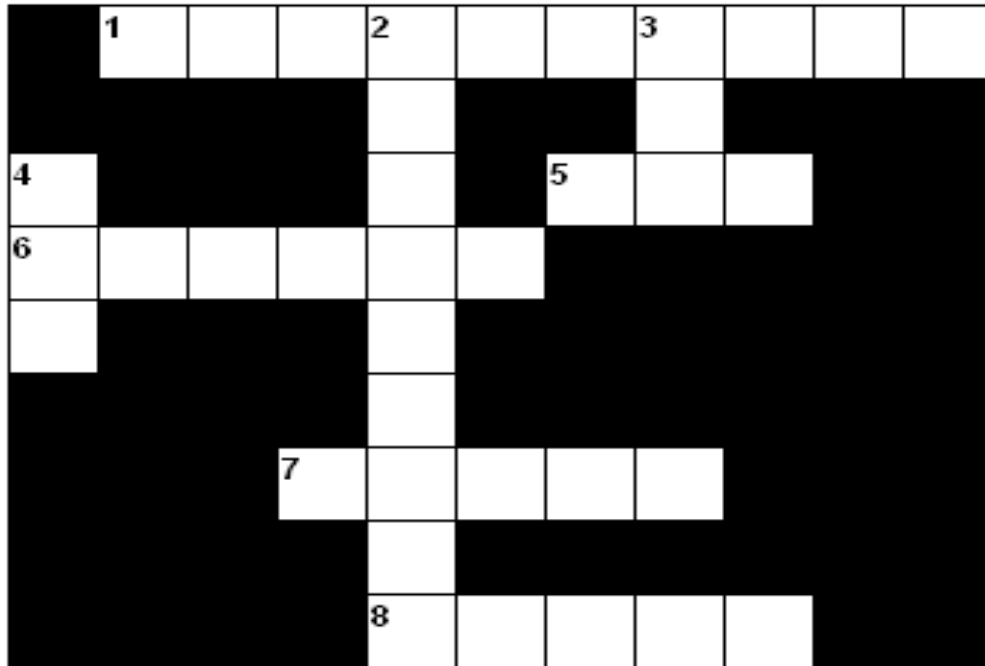
- ☑ The use of DOM and SAX parser, where to use DOM and where to use SAX.

**Best Practices :**

Use of SAX: For large documents to be loaded use SAX and for small documents use DOM.

Crossword Unit-3

Time: 15 minutes



Across:

1. _____ Parsers verify that an XML document is valid according to the rules of a DTD or schema, as well as checking for a well-formed document structure and syntax.
5. ____ is a W3C standard for accessing, navigating and manipulating XML documents.
6. The _____ reads XML into memory and converts it into an XML DOM object that can be accessed through JavaScript.
7. Compared to DOM, SAX is better at parsing _____ documents.
8. The following JavaScript fragment loads a string called txt into the parser:

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



 The first line creates a XML document object which is _____.

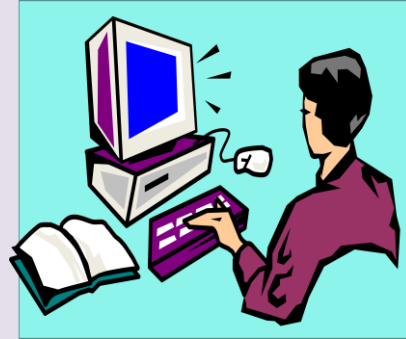
Down:

2. Document _____ is the uppermost object in the XML DOM hierarchy. It provides methods for navigating, querying and modifying the contents and structure of an XML document.
3. DOM Level ____ has extra features such as XML namespace support, filtered views, ranges and events.
4. SAX is an event-driven ____.

4.0 XML Transformations

Topics

-  4.1 Introduction to XSLT
-  4.2 Building Blocks of XSLT



Topic: Introduction to XSLT

Estimated Time: 90 mins.



Objectives : This module will familiarize the participant with

- Introduction to XSLT
- Usage of XSLT.



Presentation :

Introduction to XSLT:

- XSLT (Extensible Style Sheet Language Transformation) is a powerful tool used for transformation from XML to XML, HTML, XHTML, plain text or binary.
- It is a templating language.

Use of XSLT:

- Acts as a transformation language for XML.
- It accepts XML tree node as input and transforms them in to result using a series of templates or rules constructed.
- XSLT is based on XML syntax.
- Final output may be XML, HTML, XHTML, plain text or binary as specified by the transformation.

- **Syntax of XSLT:**

- It uses XML syntax
- The structure of XSLT has elements, functions and XPath functions
- To get access to the XSLT elements, attributes and features ,the XSLT namespace must be specified at the top of the document
- The xmlns:xsl="<http://www.w3.org/1999/XSL/Transform>" points to the official W3C XSLT namespace
- If the namespace included is <http://www.w3.org/1999/XSL/Transform> then include version="1.0" attribute.

An XML document is input for the XSLT document and the outcome will be XML, HTML, XHTML, plain text or binary. This can be clearly understood from the figure.

The comparison of XSLT with a program written in C,C# or Java

An XSLT	A program written in C, C# or Java
<ul style="list-style-type: none">• Composed of multiple templates	<ul style="list-style-type: none">• Composed of classes and modules
<ul style="list-style-type: none">• More in common with declarative languages like SQL	<ul style="list-style-type: none">• Less in common with declarative languages like SQL

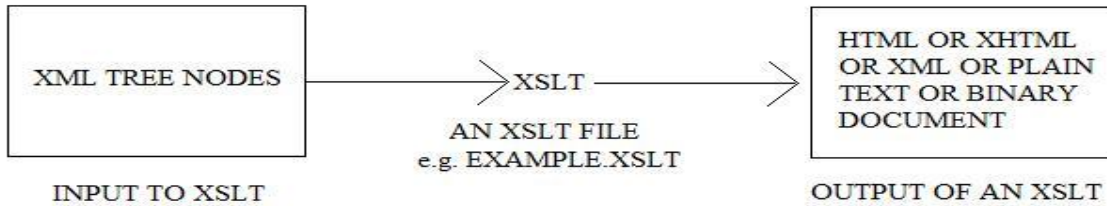


Figure 4.1-1: Input\Output of XSLT



Scenario :

Stay Connected Pvt. Ltd. is a communication equipment store selling equipments of various companies. Mobile handsets form a major chunk of communication equipment sold by Stay Connected Pvt. Ltd..

They have a customer profile of various different tastes. Whenever a customer purchases a mobile set from the store the salesman informs the customer about the additional specifications like “mp3 support” , “bluetooth” or “windows enabled mobile”. Provide a solution to Stay Connected Pvt. Ltd. With the help of XML and XSLT and generate an HTML output.



Demonstration/Code Snippet:

Step 1: Create a new XML document book.xml using visual studio 2005 or notepad.

HANDSET.XML:

```
<?xml version="1.0" encoding="utf-8"?>
<HANDSETS>
  <HANDSET>
    <COMPANY>NOKIA</COMPANY>
    <MODELNO>N91</MODELNO>
    <SOUND>MP3</SOUND>
  </HANDSET>
</HANDSETS>
```

```
<COMPANY>SONY ERICSON</COMPANY>
<MODELNO>K300</MODELNO>
<SOUND>POLYPHONIC</SOUND>
</HANDSET>
<COMPANY>SAMSUNG</COMPANY>
<MODELNO>R220</MODELNO>
<SOUND>MONOPHONIC</SOUND>
</HANDSET>
</HANDSETS>
```



Knowledge of XML: You should have knowledge of XML and XPath expression to understand XSLT.

Step 2: Create an XSLT file in notepad by name “HNDSET.XSLT”.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h2>MOBILES</h2>
        <table border="1">
          <tr>
            <th>COMPANY</th>
            <th>MODEL</th>
            <th>SOUND</th>
          </tr>
          <xsl:for-each select="HANDSETS/HANDSET">
            <tr>
              <td>
            </td>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Out-put:

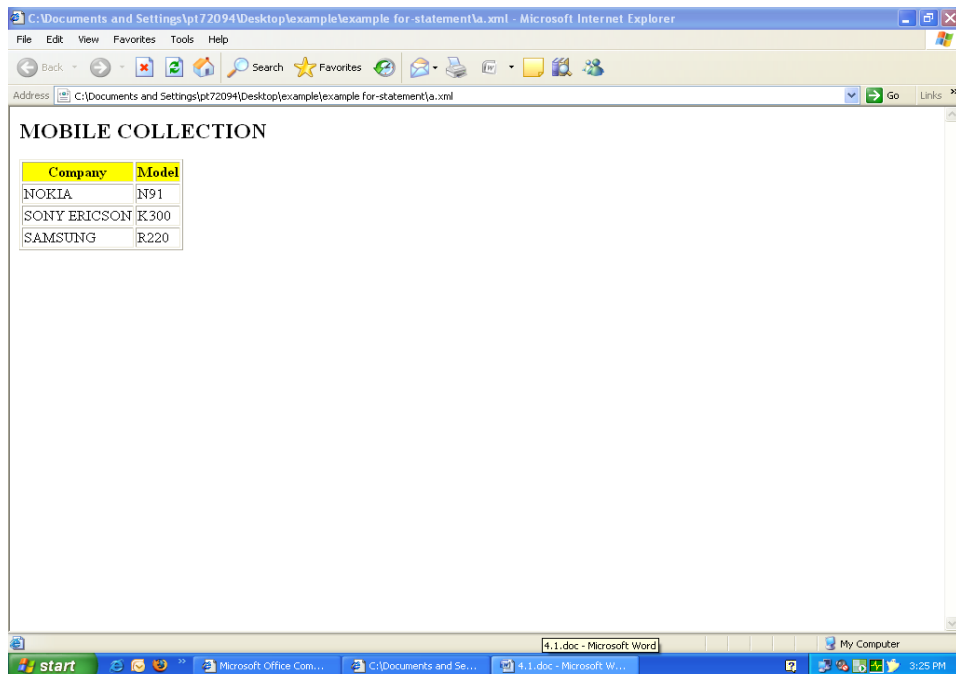


Figure 4.1-2: Output



Context:

- Use of an XSLT to retrieve the data from an XML document in a required format.



Practice Session:

- Create an XSLT file to retrieve required handset data by applying it on HANDSET.XML file.



Check list:

- XSLT transformation elements



Common Errors:

- Syntax not properly defined.
- Element starts with xsl prefix then name and type of element.



Lessons Learnt:

- ☒ XSLT and how to use it.

**Best Practices:**

- XSLT is an efficient standard to retrieve data from an XML document.

**Topic: Building Blocks of an XSLT****Estimated Time: 150 Mins.****Objectives :** This module will familiarize the participant with

- The different building blocks of XSLT
- Use of template element within an XSLT.

**Presentation :**

- The purpose of XSLT is to retrieve data from an XML document.
- To retrieve data from an XML, blocks are written within an XSLT document
- The building blocks of an XSLT are not similar to normal languages like C, C# and Java, so it is very important to understand the building blocks of an XSLT

Elements of an XSLT document:

- stylesheet or transform
- import, include
- strip-space, preserve-space
- decimal-format
- output
- key
- variable
- param
- template



Note: All elements do not work in internet explorer current version. It would either work in higher version or in another browser..



Presentation:

Stylesheet or transform

- It is a root node for XSLT document
- Namespace used by this tag is <http://www.w3.org/1999/XSL/Transform>
- 'transform' acts as a synonym for 'stylesheet'
- Either 'transform' or 'stylesheet' uses a single attribute 'version'
- "1.0" or "2.0" are the only valid values for attribute 'version'



Demonstration/Code Snippet :

Example code 'stylesheet' element:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
.....
</xsl:stylesheet>
```

Example code 'transform' element:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
.....
</xsl:transform>
```



Presentation:

Import, include:

import:

- Imports one stylesheet in to another
- If 'import' element is used in our stylesheet, it must be the first node after the root node
- It helps to enable the modularization within stylesheet
- If an element exists in parent as well as in imported stylesheet, element in parent stylesheet takes the precedence
- It means during processing a stylesheet lower precedence elements are ignored in favor of higher ones

Include:

- 'include' acts in same fashion as 'import' tag with one exception
- While using 'include' tag the stylesheet being included and the parent stylesheet have the same precedence
- So if there are duplicate element, variables or templates, they can lead to error

Both 'import' and 'include' have a single attribute 'href' that points to the imported or included stylesheet



Demonstration/Code Snippet :

Example code for 'import' element:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  <xsl:import href="importme.xslt"/>
  .....
</xsl:stylesheet>
```

Example code for 'include' element:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:include href="importme.xslt"/>
  .....
</xsl:stylesheet>
```



Presentation:

Strip-space, preserve-space:

- strip-space
 - 'strip-space' element shows number of elements for which the white space has been removed.
- preserve-space
 - 'preserve-space' element shows number of elements for which the white space has been preserved during processing
- It is to be noted that the elements will affect only those text nodes which contains white spaces, not all the nodes present within a document.
- 'strip-space' and 'preserve-space' both the elements have a common attribute called 'elements'
- Here the attribute 'elements' can have either space delimited list of elements or all the element using '*' symbol



Demonstration/Code Snippet :

Syntax for strip-space and preserve-space element:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="utf-8" indent="yes"/>
  <xsl:preserve-space elements="COMPANY"/>
  <xsl:strip-space elements="SOUND"/>
  .....
</xsl:stylesheet>
```



Presentation :

Decimal-format:

- This element describes the symbols to be used when formatting the numbers using XSLT
- The rules constructed to format a number using 'decimal-format' element can be applied using the 'format-number' function
- Example: In India "ten thousand rupees" is written in a format 8,000; while in U.S. the same is written in a format: 80,00

Along with this, decimal-format function is used to change the value being displayed if the value that is input is not a number, infinity or negative.



Presentation :

Output:

- The <xsl:output> element defines the format of the output document
- It is a top-level element, and must appear as a child node of <xsl:stylesheet> or <xsl:transform>

Attribute	Value	Description
✓ Method	xml html text name	Optional. Defines the output format. The default is XML (but if the first child of the root node is <html> and there are no preceding text nodes, then the default is HTML) Netscape 6 only supports "html" and "xml"
✓ Version	String	Optional. Sets the W3C version number for the output format (only used with method="html" or method="xml")
✓ encoding	String	Optional. Sets the value of the encoding attribute in the output
✓ omit-xml-declaration	yes no	Optional. "yes" specifies that the XML declaration (<?xml...?>) should be omitted in the output. "no" specifies that the XML declaration should be included in the output. The default is "no"
✓ standalone	yes no	Optional. "yes" specifies that a standalone declaration should occur in the output. "no" specifies that a standalone declaration should not occur in the output. The default is "no" This attribute is not supported by Netscape 6
✓ doctype-public	String	Optional. Sets the value of the PUBLIC attribute of the DOCTYPE declaration in the output
✓ doctype-system	String	Optional. Sets the value of the SYSTEM attribute of the DOCTYPE declaration in the output
✓ cdata-section-elements	namelist	Optional. A white-space separated list of elements whose text contents should be written as CDATA sections
✓ indent	yes no	Optional. "yes" indicates that the output should be indented according to its hierarchic structure. "no" indicates that the

		output should not be indented according to its hierarchic structure.
		This attribute is not supported by Netscape 6
✓ media-type	String	Optional. Defines the MIME type of the output. The default is "text/xml"
		This attribute is not supported by Netscape 6



Demonstration/Code Snippet :

An Example code of 'output' element to set output as XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="utf-8" indent="yes"/>
  .....
</xsl:stylesheet>
```

An Example code of 'output' element to set output as XML format:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" version="4.0" encoding="utf-8" indent="yes"/>
  .....
</xsl:stylesheet>
```



Presentation:

Key:

- A named key is being created using element 'key'
- This named key is very useful while searching a source document for a specific record
- In An XML file an element can be uniquely identified using an identifier often know as @id
- In same manner 'key' element identifies element using attribute or child element as an identifier for each element
- This technique is also useful with XSLT for retrieving individual items



Demonstration/Code Snippet :

Syntax to create a named key:

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="utf-8" indent
="yes"/>
  <xsl:preserve-space elements="COMPANY"/>
  <xsl:strip-space elements="SOUND"/>
  .....
</xsl:stylesheet>
```



Presentation:

Template:

- The core concept of an XSLT is template
- It is similar to the sub-routine or function of the languages like C, C#, Java
- Look at the structure of an template, it is composed of ‘template’ element along with ‘match’ and/or ‘name’ attribute
- ‘match’ is the common attribute which identifies XPath pattern in source document to which this template is applicable
- ‘name’ attribute is used only in a case when call template is used
- You can visualize the output of template by examining the content of a template

A user-defined function in C language:

```
int sumnumbers(int d_first,int d_second)
{
    int temp_result;
    temp_result=d_first+d_second;
    return temp_result;
}
```

A template example for XSLT:

```
<xsl:template match="HANDSET">
```

```
<HNDST>
  <xsl:attribute name="MODELNO">
    <xsl:value-of select="SOUND"/>
  </xsl:attribute>
</HNDST>
</xsl:template>
```



Presentation:

<xsl:applytemplate>

- <xsl:applytemplate> applies a template (functional block) to the current element or to the current element's child nodes
- While selecting attribute using the <xsl:applyattribute>, it processes childnode that matches the value of the attribute
- If 'select' attribute is removed, child node in a document are processed in order they appear in a source document



Presentation:

<xsl:calltemplate>

- Executes a named template based on the current context
- It is similar to calling a function
- It is mainly used with conditional logic and/or parameters



Presentation:

Variable:

- It is used to declare a variable within an XSLT
- If it is declared within a template then it is known as local variable
- If it is declared as a top level element then it is known as global variable
- The main difference between <xsl:param> and <xsl:variable> is that once we declare value for <xsl:variable> we cannot change or modify the value

Attribute	Value	Description
✓ Name	name	Required. Specifies the name of the variable
✓ Select	expression	Optional. Defines the value of the variable

Syntax for variable element:

```
<xsl:variable name="color" select="'red'" />
```



Presentation:

param:

- <xsl:param> is used to declare a variable within an XSLT document
- If a variable is declared within <xsl:template> tag then it is treated as a local variable
- If a variable is declared using <xsl:param> as a top level element then it is treated as a global variable

Attribute	Value	Description
✓ Name	name	Required. Specifies the name of the parameter
✓ select	expression	Optional. Specifies an XPath expression that specifies a default value for the parameter



Demonstration/Code Snippet:

```
<xsl:call-template name="show_title">
<xsl:with-param name="title" />
</xsl:call-template>
</body>
</html>
</xsl:variable>
<xsl:template name="show_title" match="/">
  <xsl:param name="title" />
  <xsl:for-each select="COMPANY">
    <p>
      Title: <xsl:value-of select="$title" />
    </p>
  </xsl:for-each>
</xsl:template>
```



Presentation:

<xsl:valueof> element:

- This element is used to extract the value of an XML element

- The extracted value is added to the output stream of the transformation



Demonstration/Code Snippet:

```
<xsl:value-of select="HANDSETS/HANDSET/COMPANY"/>
```



Presentation:

<xsl:for-each>

- <xsl:for-each> element can be used to select every XML element of a specified node-set



Demonstration/Code Snippet :

```
<xsl:for-each select="HANDSETS/HANDSET">
  <tr>
    <td>
      <xsl:value-of select="COMPANY"/>
    </td>
    <td>
      <xsl:value-of select="MODELNO"/>
    </td>
  </tr>
</xsl:for-each>
```



Presentation :

<xsl:if>

- Along with looping we also have provision for conditional test within XSLT



Demonstration/Code Snippet :

```
<xsl:if test="SOUND = MP3">
  <tr>
    <td>
      <xsl:value-of select="COMAPNY"/>
    </td>
    <td>
      <xsl:value-of select="MODELNO"/>
    </td>
  </tr>
</xsl:if>
```

```
</td>
</tr>
</xsl:if>
```



Presentation:

<xsl:sort>

- Used to provide the sorted transformation result



Demonstration/Code Snippet :

```
<xsl:sort select="COMAPNY"/>
```



Presentation :

<xsl:choose>

- <xsl:choose> element is used in conjunction with <xsl:when> and <xsl:otherwise>
- <xsl:choose> is very useful to provide multiple condition tests



Demonstration/Code Snippet :

```
<xsl:choose>
  <xsl:when test="SOUND=MP#">
    <td bgcolor="#ff00ff">
      <xsl:value-of select="MODELNO"/>
    </td>
  </xsl:when>
  <xsl:otherwise>
    <td>
      <xsl:value-of select="MODELNO"/>
    </td>
  </xsl:otherwise>
</xsl:choose>
```



Scenario :

ConnectMe Pvt Ltd is a communication equipment store. The communication equipment that they sell most in an Indian market are the mobile handsets. On customer demand they mail the list of mobile handsets also. Provide a solution to ConnectMe Pvt. Ltd to maintain a database that has information about the mobile handsets. The information should be easy to transfer and be in a readable format for the customer independent of the reader used by him.



Demonstration/Code Snippet :

Step 1: As we know that we need a solution to store data in such a way that it should be independent of text reader used by the user. For this purpose we must take help of XML. So create an XML document using notepad.

Start menu → Run menu → notepad → SaveAs XML (Details.XML)

```
<?xml versiontype="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="handset.xslt"?>
<HANDSETS>
  <HANDSET>
    <COMPANY>NOKIA</COMPANY>
    <MODELNO>N91</MODELNO>
    <SCREEN COLOR="yes"/>
    <CONNECTIVITY BLUETOOTH="yes" INFRARED="no"/>
    <SOUND>MP3</SOUND>
    <NETWORK>GSM</NETWORK>
  </HANDSET>

  <HANDSET>
    <COMPANY>SAMSUNG</COMPANY>
    <MODELNO>C170</MODELNO>
    <SCREEN COLOR="no"/>
    <CONNECTIVITY BLUETOOTH="no" INFRARED="no"/>
    <SOUND>POLYPHONIC</SOUND>
    <NETWORK>GSM</NETWORK>
  </HANDSET>

  <HANDSET>
    <COMPANY>SONY ERICSON</COMPANY>
    <MODELNO>Z550i</MODELNO>
    <SCREEN COLOR="yes"/>
    <CONNECTIVITY BLUETOOTH="yes" INFRARED="yes"/>
    <SOUND>MP3</SOUND>
    <NETWORK>GSM</NETWORK>
  </HANDSET>

  <HANDSET>
    <COMPANY>MOTOROLA</COMPANY>
    <MODELNO>E6</MODELNO>
    <SCREEN COLOR="yes"/>
    <CONNECTIVITY BLUETOOTH="yes" INFRARED="no"/>
    <SOUND>MP3</SOUND>
    <NETWORK>GSM</NETWORK>
  </HANDSET>

  <HANDSET>
    <COMPANY>SAMSUNG</COMPANY>
```



```
<MODELNO>R220</MODELNO>
<SCREEN COLOR="no"/>
<CONNECTIVITY BLUETOOTH="no" INFRARED="no"/>
<SOUND>MONOPHONIC</SOUND>
<NETWORK>CDMA</NETWORK>
</HANDSET>
```

```
<HANDSET>
<COMPANY>SAMSUNG</COMPANY>
<MODELNO>X100</MODELNO>
<SCREEN COLOR="yes"/>
<CONNECTIVITY BLUETOOTH="no" INFRARED="yes"/>
<SOUND> </SOUND>
<NETWORK> </NETWORK>
</HANDSET>
```

```
<HANDSET>
<COMPANY>SONY ERICSON</COMPANY>
<MODELNO>K500i</MODELNO>
<SCREEN COLOR="yes"/>
<CONNECTIVITY BLUETOOTH="no" INFRARED="yes"/>
<SOUND>MP3</SOUND>
<NETWORK>GSM</NETWORK>
</HANDSET>
```

```
<HANDSET>
<COMPANY>MOTOROLA</COMPANY>
<MODELNO>V6</MODELNO>
<SCREEN COLOR="yes"/>
<CONNECTIVITY BLUETOOTH="yes" INFRARED="no"/>
<SOUND>MP3</SOUND>
<NETWORK>GSM</NETWORK>
</HANDSET>
```

```
<HANDSET>
<COMPANY>NOKIA</COMPANY>
<MODELNO>1600</MODELNO>
<SCREEN COLOR="no"/>
<CONNECTIVITY BLUETOOTH="no" INFRARED="no"/>
<SOUND>MONOPHONIC</SOUND>
<NETWORK>GSM</NETWORK>
</HANDSET>
```

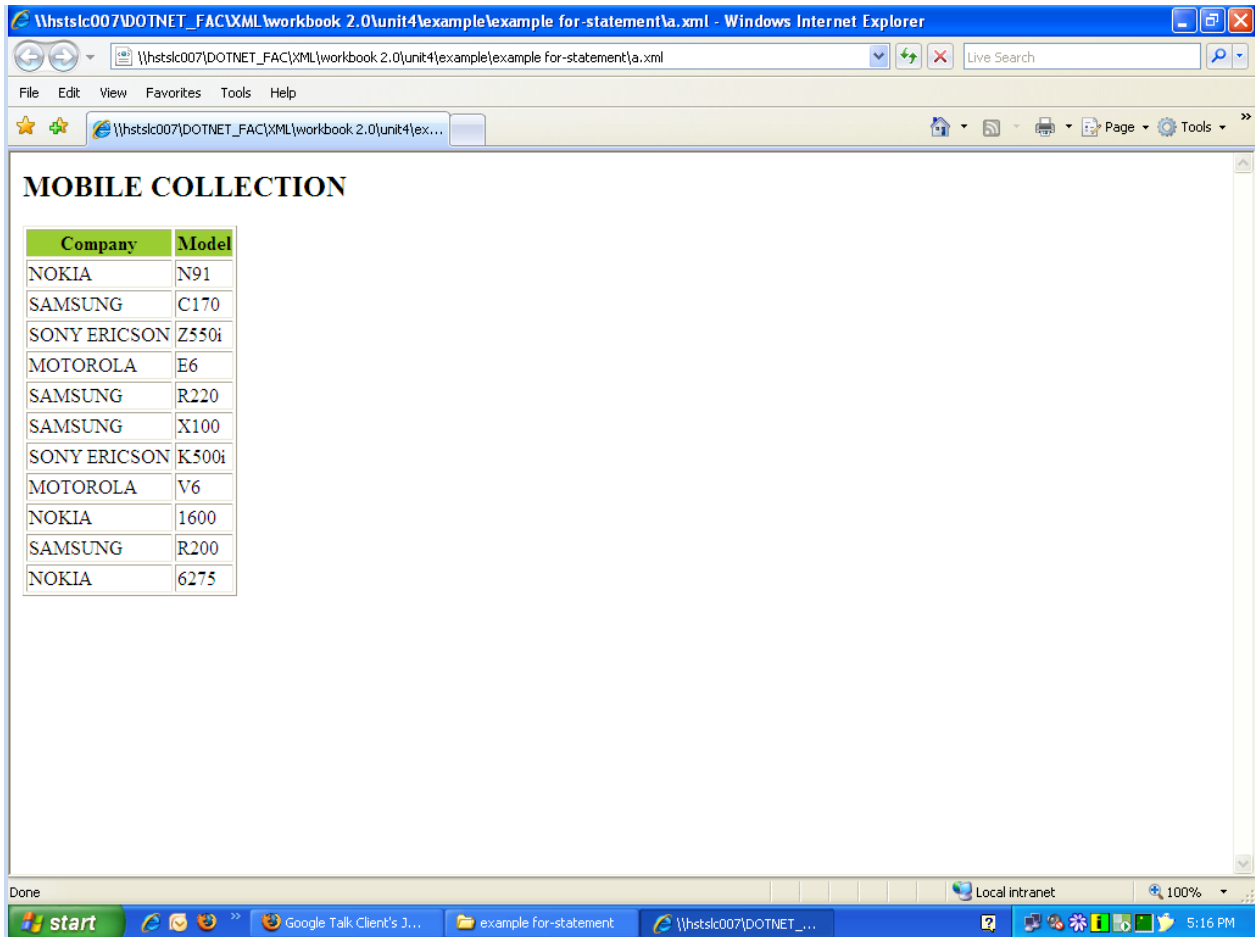
```
<HANDSET>
<COMPANY>SAMSUNG</COMPANY>
<MODELNO>R200</MODELNO>
<SCREEN COLOR="no"/>
<CONNECTIVITY BLUETOOTH="no" INFRARED="no"/>
<SOUND>MONOPHONIC</SOUND>
<NETWORK>CDMA</NETWORK>
</HANDSET>
```

```
<HANDSET>
<COMPANY>NOKIA</COMPANY>
<MODELNO>6275</MODELNO>
<SCREEN COLOR="yes"/>
<CONNECTIVITY BLUETOOTH="yes" INFRARED="yes"/>
```

```
<SOUND>MP3</SOUND>
<NETWORK>CDMA</NETWORK>
</HANDSET>
</HANDSETS>
```

Step 2: Now to provide better readability to user we will provide an XSLT along with XML file which can generate XML, HTML, XHTML, plain text or binary output. Here we are providing HTML output. Start menu → Run menu → notepad → SaveAs XML (Handset.XSLT)

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0" encoding="utf-8" indent="yes"/>
<xsl:template match="/">
<html>
  <body>
    <h2>MOBILE MODELS</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>COMPANY</th>
        <th>MODELNO</th>
      </tr>
<xsl:for-each select="HANDSETS/HANDSET[COMAPNY='SONY ERICSON']">
      <tr>
        <td>
          <xsl:value-of select="COMAPNY"/>
        </td>
        <td>
          <xsl:value-of select="MODLES"/>
        </td>
      </tr>
    </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```



The screenshot shows a Windows Internet Explorer browser window. The title bar reads "Whstslc007\DOTNET_FAC\XML\workbook 2.0\unit4\example\example for-statement\va.xml - Windows Internet Explorer". The address bar shows the file path: "\\hstslc007\DOTNET_FAC\XML\workbook 2.0\unit4\example\example for-statement\va.xml". The browser displays a table titled "MOBILE COLLECTION". The table has two columns: "Company" and "Model". The data rows are as follows:

Company	Model
NOKIA	N91
SAMSUNG	C170
SONY ERICSON	Z550i
MOTOROLA	E6
SAMSUNG	R220
SAMSUNG	X100
SONY ERICSON	K500i
MOTOROLA	V6
NOKIA	1600
SAMSUNG	R200
NOKIA	6275

The browser window also shows a taskbar at the bottom with the Start button, several open applications (Google Talk Client's J..., example for-statement, Whstslc007\DOTNET_...), and a system tray with the time 5:16 PM and date 5/16/2007.

Figure 4.2-1: Output



Knowledge of XML: You should have knowledge of XML to understand XSLT.



Context :

- How to construct an XSLT and use different building blocks to increase the programmability.



Practice Session :

- Geezmogames Pvt. Ltd. is an entertainment company. . Computer games are the main products of Geezmogames Pvt. Ltd. Along with the CD of the game they provide a list of computer resources required to play the game for e.g. graphics card, RAM, Screen resolution, operating system etc. After that they received some complaints. The format of resource document was not compatible with different text editors used by customers. . Provide a format to Geezmogames Pvt.Ltd. which is compatible with various text editors or web browsers. The resources mentioned in a document are Operating System along with version and patches released, Graphics Card with capacity, RAM with size, Monitor with screen resolution specification, processor required (showing minimum requirement e.g. minimum P-III),HDD mentioning capacity and optical drive required for installation.
- JeevanSangeet Pvt. Ltd. is an audio company. They are launching MP3 which contains the collection of songs sung by Kishore Kumar. They provide a list of songs for the computer users. Suggest a format which is compatible with web browsers like Internet Explorer or Firefox etc. Also provide a solution which enlists songs sung by Kishore Kumar and music given by R.D.Burman. The data contain Song title, Singer's name, Time duration, musician, Date on which recorded, whether song is solo or duet, information about chorus, film or album to which the song belongs.



Check List :

- ☒ Introduction to various building blocks of an XSLT.
- ☒ Understanding loop statement and conditional statements within XSLT.
- ☒ How to create templates and call templates within an XSLT.



Common Errors :

- Forget to add “`<?xml-stylesheet type="text/xsl" href="handset.xslt" ?>`” within an XML document to be parsed.



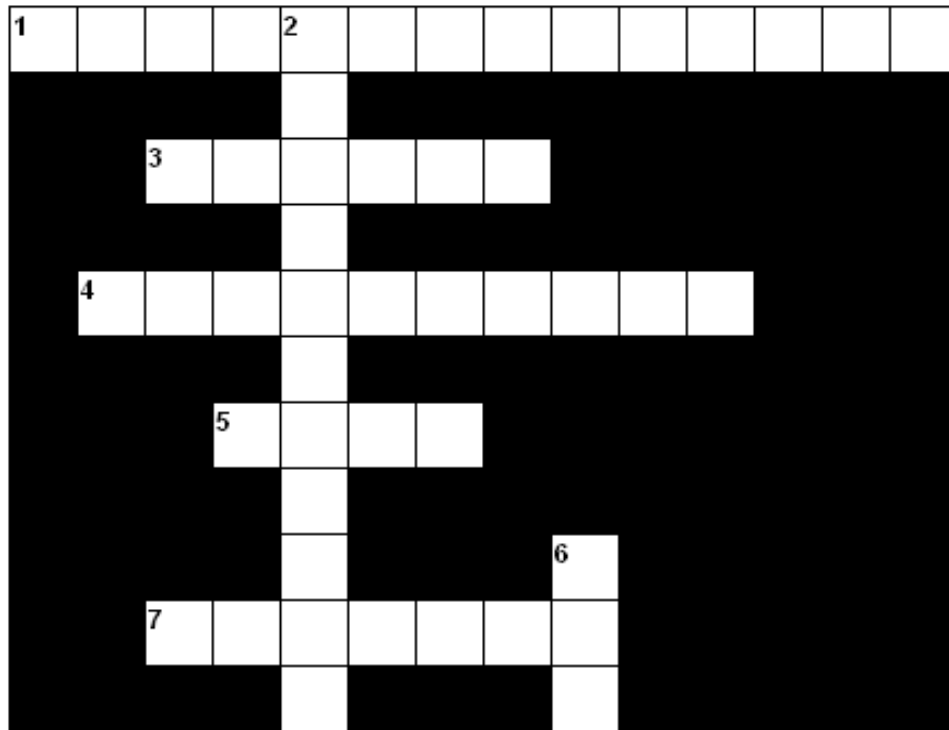
Lessons Learnt :

- How to use loop statements, conditional statements and functions within an XSLT and retrieve data more efficiently.



Best Practices:

- Use templates within an XSLT and reuse them as and when needed.

CROSSWORD UNIT 4
Time: 20 minutes

Across:



1. XSLT accepts XML tree node as input and transforms them in to result using a series of templates or rules constructed. Hence XSLT acts as a _____ language for XML.
3. If an element exists in parent as well as imported stylesheet, which one takes precedence?
4. XSLT is a _____ language.
5. _____ is a powerful tool used for transformation from XML to XML, HTML, XHTML, plain text or binary.
7. Which term is used to give equal precedence to both the stylesheet being included and the parent stylesheet?

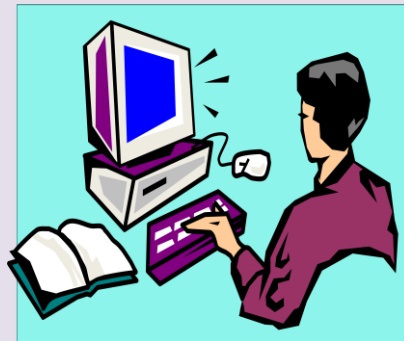
Down:

2. _____ element shows number of elements for which the white space has been removed.
6. A named key is being created using element _____.

5.0 Querying XML

Topics

-  5.1 XPath
-  5.2 XQuery



Topic: XPath

Estimated Time: 150 Mins.



Objectives : This module will familiarize the participant with

- Introduction to XPath
- Use of XPath.
- Example Of XPath



Demonstration/Code Snippet :

We will consider the following XML file as an example code for all the upcoming example.

```
<?xml version ="1.0" encoding ="utf-8"?>
<HANDSETS>
  <HANDSET type ="GSM">
    <COMPANY>SAMSUNG</COMPANY>
    <MODELNO>C170</MODELNO>
    <SOUND>POLYPONIC</SOUND>
  </HANDSET>
  <HANDSET type ="GSM">
    <COMPANY>NOKIA</COMPANY>
    <MODELNO>N91</MODELNO>
    <SOUND>MP3</SOUND>
  </HANDSET>
  <HANDSET type ="CDMA">
    </COMPANYMODELNO>SONY ERICSON</COMPANY>
    <MODELNO></MODELNO>
    <SOUND> MP3</SOUND>
  </HANDSET>
  <HANDSET type ="GSM">
    <COMPANY>SAMSUNG</COMPANY>
    <MODELNO>R220</MODELNO>
    <SOUND> </SOUND>
  </HANDSET>
</HANDSETS>
```



Presentation :

Introduction to XPath

- XPath is a syntax for defining parts of an XML document
- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions
- XPath is a major element in XSLT

- XPath is a W3C Standard

XPath is a language for finding information in an XML document. XPath is used to navigate through elements and attributes in an XML document.



Presentation :

XPath Path Expressions

- Path expressions are used to select the nodes and nodes-set from an xml document
- These expressions are very similar to traditional file-system expressions



Presentation :

XPath standard functions

- XPath includes over 100 built-in functions
- Functions are available for manipulations over string values, numeric values, date and time comparison

Along with that functions are available for manipulation over node and QName manipulation, sequence manipulation, Boolean values, and more



Presentation :

XPath Nodes

There are seven different types of nodes:

- Element
- Attribute
- Text
- Namespace
- Processing-instruction
- Comment
- Document (root)



Demonstration/Code Snippet :

Type of nodes available in above docuemnt:

- `<HANDSETS>` → Document node
- `<COMPANY>NOKIA</COMPANY>` → Element node
- `<HANDSET type = "GSM">` → Attribute node



Presentation:

Atomic Values:

- Atomic values are values with no child or parent e.g. NOKIA, "GSM"



Presentation :

Relationship of Nodes:

- **Parent:**

Each element and attribute has one parent

For the following example code the parent is `<HANDSET>`

```
<HANDSET type = "GSM">
  <COMPANY>SAMSUNG</COMPANY>
  <MODELNO>C170</MODELNO>
  <SOUND>POLYPONIC</SOUND>
</HANDSET>
```

- **Children:**

Element node may have zero, one or more child.

For the following code COMPANY, MODELNO AND SOUND are children of HANDSET element.

```
<HANDSET type = "GSM">
  <COMPANY>SAMSUNG</COMPANY>
  <MODELNO>C170</MODELNO>
  <SOUND>POLYPONIC</SOUND>
</HANDSET>
```

- **Ancestors**

An ancestor may be a node's parent, parent's parent, etc.

For the following code HANDSET element is the ancestor for COMPANY, SOUND as well as MODELNO element.

```
<HANDSET type ="GSM">
  <COMPANY>SAMSUNG</COMPANY>
  <MODELNO>C170</MODELNO>
  <SOUND>POLYPONIC</SOUND>
</HANDSET>
```

- **Descendants**

A descendent may be a node's children, children's children, etc.

For the following code COMPANY, MODEL NO and SOUND are the child nodes for HANDSET ELEMENT.

```
<HANDSET type ="GSM">
  <COMPANY>SAMSUNG</COMPANY>
  <MODELNO>C170</MODELNO>
  <SOUND>POLYPONIC</SOUND>
</HANDSET>
```



Presentation:

XPath Syntax

- Selecting Nodes
 - Path expressions are used to select nodes within an XML document
 - Node is being selected by following path or steps

The table shown below explains some of the very useful path expressions.

Expression	Description
<i>Nodename</i>	Selects all child nodes of the named node
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

To understand the table shown below consider the above code.

Path Expression	Description	Result
HANDSETS	Selects all the child nodes of the HANDSETS element	<COMPANY>, <MODELNO> and <SOUND> will be selected as a result of these.
/HANDSETS	Selects the root element HANDSETS	<HANDSETS>
	Note: If the path starts with a slash </HANDSET> element will be

	'/' it always represents an absolute path to an element!	selected.
HANDSETS/HANDSET	Selects all HANDSET elements that are children of HANDSETS	All three <HANDSET>...<HANDSET> elements will be selected.
//HANDSET	Selects all HANDSET elements no matter where they are in the document	All three <HANDSET>...<HANDSET> elements will be selected.
HANDSETS//HANDSET	Selects all HANDSET elements that are descendant of the HANDSETS element, no matter where they are under the HANDSETS element	All three <HANDSET>...<HANDSET> elements will be selected.
//@type	Selects all attributes that are named type	Attribute type="GSM" will be selected.

- **Predicates**

- Predicates are always embedded in square brackets.
- Predicates are used to find a specific node or a node that contains a specific value.

To understand the table shown below please consider the code given here:

Path Expression	Description	Result
/HANDSETS/HANDSET[1]	Selects the first HANDSET element that is the child of the HANDSETS element. Note: IE5 and later has implemented that [0] should be the first node, but according to the W3C standard it should have been [1]!!	<COMPANY>SAMSUNG</COMPANY> <MODELNO>C170</MODELNO> <SOUND>POLYPONIC</SOUND>
/HANDSETS/HANDSET[last()]	Selects the last HANDSET element that is the child of the HANDSETS element	<COMPANY>SAMSUNG</COMPANY> <MODELNO>R220</MODELNO> <SOUND> </SOUND>
/HANDSETS/ HANDSET [last()-1]	Selects the last but one HANDSET element that is the child of the HANDSETS element	<COMPANY>SAMSUNG</COMPANY> <MODELNO>R220</MODELNO> <SOUND> </SOUND>
/ HANDSETS/ HANDSET[position()<3]	Selects the first two HANDSET elements that are children of the HANDSETS element	<HANDSET type ="GSM"> <COMPANY>SAMSUNG</COMPANY> <MODELNO>C170</MODELNO> <SOUND>POLYPONIC</SOUND> </HANDSET> <HANDSET type ="GSM">

		<pre><COMPANY>NOKIA</COMPANY> <MODELNO>N91</MODELNO> <SOUND>MP3</SOUND> </HANDSET></pre>
//title[@type]	Selects all the title elements that have an attribute named type	<pre><HANDSET type ="GSM"> .. </ HANDSET > <HANDSET type ="GSM"> .. </ HANDSET > <HANDSET type ="CDMA"> .. </ HANDSET > <HANDSET type ="GSM"> .. </ HANDSET ></pre>
//title[@type='GSM']	Selects all the title elements that have an attribute named type with a value of 'GSM'	<pre><HANDSET type ="GSM"> .. </ HANDSET > <HANDSET type ="GSM"> .. </ HANDSET > <HANDSET type ="GSM"> .. </ HANDSET ></pre>



Presentation :

Selecting unknown nodes

- XPath wildcards can be used to select unknown XML elements

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

Path Expression	Description	Result
/HANDSET/*	Selects all the child nodes of the HANDSET element	<pre><COMPANY>SAMSUNG</COMPANY> <MODELNO>C170</MODELNO> <SOUND>POLYPONIC</SOUND></pre>
//*	Selects all elements in the document	<pre><HANDSETS> <HANDSET type ="GSM"> <COMPANY>SAMSUNG</COMPANY> <MODELNO>C170</MODELNO> <SOUND>POLYPONIC</SOUND> </HANDSET> <HANDSET type ="GSM"> <COMPANY>NOKIA</COMPANY> <MODELNO>N91</MODELNO></pre>

		<pre> <SOUND>MP3</SOUND> </HANDSET> <HANDSET type ="CDMA"> <COMPANY>SONY ERICSON</COMPANY> <MODELNO></MODELNO> <SOUND> MP3</SOUND> </HANDSET> <HANDSET type ="GSM"> <COMPANY>SAMSUNG</COMPANY> <MODELNO>R220</MODELNO> <SOUND>POLYPONIC</SOUND> </HANDSET> </HANDSETS> </pre>
//title[@*]	Selects all title elements which have any attribute	<pre> <HANDSET type ="GSM"> </HANDSET> <HANDSET type ="GSM"> </HANDSET> <HANDSET type ="CDMA"> </HANDSET> <HANDSET type ="GSM"> </HANDSET> </pre>



Presentation :

Selecting several paths

- By using the | operator in an XPath expression you can select several paths

Path Expression	Description	Result
//HANDSET/COMPANY //HANDSET/MODELNO	Selects all the COMPANY AND MODELNO elements of all HANDSET elements	<pre> <COMPANY>SAMSUNG</COMPANY> <MODELNO>C170</MODELNO> <COMPANY>NOKIA</COMPANY> <MODELNO>N91</MODELNO> <COMPANY>SONYERICSON</COMPANY> <MODELNO></MODELNO> <COMPANY>SAMSUNG</COMPANY> <MODELNO>R220</MODELNO> </pre>
//COMAPNY //MODELNO	Selects all the COMPANY AND MODELNO elements in the document	<pre> <COMPANY>SAMSUNG</COMPANY> <MODELNO>C170</MODELNO> <COMPANY>NOKIA</COMPANY> <MODELNO>N91</MODELNO> <COMPANY>SONYERICSON</COMPANY> <MODELNO></MODELNO> <COMPANY>SAMSUNG</COMPANY> <MODELNO>R220</MODELNO> </pre>
/HANDSETS/HANDSET/COMPANY //MODELNO	Selects all the title elements of the HANDSET element of the HANDSETS element AND all the MODELNO elements in the	<pre> <COMPANY>SAMSUNG</COMPANY> <MODELNO>C170</MODELNO> <COMPANY>NOKIA</COMPANY> <MODELNO>N91</MODELNO> <COMPANY>SONYERICSON</COMPANY> <MODELNO></MODELNO> <COMPANY>SAMSUNG</COMPANY> <MODELNO>R220</MODELNO> </pre>

	document	
--	----------	--



Presentation :

XPath Operators

Operator	Description	Example	Return value
	Computes two node-sets	//book //cd	Returns a node-set with all book and cd elements
+	Addition	6 + 4	10
-	Subtraction	6 - 4	2
*	Multiplication	6 * 4	24
Div	Division	8 div 4	2
=	Equal	price=9.80	true if price is 9.80 false if price is 9.90
!=	Not equal	price!=9.80	true if price is 9.90 false if price is 9.80
<	Less than	price<9.80	true if price is 9.00 false if price is 9.80
<=	Less than or equal to	price<=9.80	true if price is 9.00 false if price is 9.90
>	Greater than	price>9.80	true if price is 9.90 false if price is 9.80
>=	Greater than or equal to	price>=9.80	true if price is 9.90 false if price is 9.70
Or	or	price=9.80 or price=9.70	true if price is 9.80 false if price is 9.50
And	and	price>9.00 and price<9.90	true if price is 9.80 false if price is 8.50
Mod	Modulus (division remainder)	5 mod 2	1



Presentation:

XPath Axes

- A node-set relative to current node is known as 'Axes'

Axis Name	Result
Ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
Attribute	Selects all attributes of the current node
Child	Selects all children of the current node

Descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
Following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
Namespace	Selects all namespace nodes of the current node
Parent	Selects the parent of the current node
Preceding	Selects everything in the document that is before the start tag of the current node
preceding-sibling	Selects all siblings before the current node
Self	Selects the current node



Presentation :

Location Path Expression

- Location path can be absolute or relative
- An absolute location path starts with a slash '/'
- Example of an absolute path is '/step/step/'
- A relative location path does not start with '/'
- Example of an relative path is 'step/step/'
- For both cases the location path consists of one or more steps separated by a slash
- Each step is evaluated against the nodes in the current node-set

A step consists of:

- an axis (defines the tree-relationship between the selected nodes and the current node)
- a node-test (identifies a node within an axis)
- zero or more predicates (to further refine the selected node-set)
- Syntax for a location step is 'axisname::nodetest[predicate]'
-

Example	Description	Result
child::HANDSET	Selects all HANDSET nodes that are children of the current node	<pre> <COMPANY>SAMSUNG</COMPANY> <MODELNO>C170</MODELNO> <SOUND>POLYPONIC</SOUND> </HANDSET> <HANDSET type ="GSM"> <COMPANY>NOKIA</COMPANY> <MODELNO>N91</MODELNO> <SOUND>MP3</SOUND> </HANDSET> <HANDSET type ="CDMA"> <COMPANY>SONYERICSON</COMPANY> <MODELNO> K300</MODELNO> <SOUND> MP3</SOUND> </HANDSET> </pre>

		<code></HANDSET></code> <code><HANDSET type = "GSM"></code> <code><COMPANY>SAMSUNG</COMPANY></code> <code><MODELNO>R220</MODELNO></code> <code><SOUND>POLYPONIC</SOUND></code> <code></HANDSET></code>
attribute::type	Selects the type attribute of the current node	<code>type = "GSM"</code>
child::*	Selects all children of the current node	<code><COMPANY>SAMSUNG</COMPANY></code> <code><MODELNO>R220</MODELNO></code> <code><SOUND>POLYPONIC</SOUND></code>
attribute::*	Selects all attributes of the current node	<code>type = "CDMA"</code>
child::text()	Selects all text child nodes of the current node	<code><COMPANY>SAMSUNG</COMPANY></code> <code><MODELNO>R220</MODELNO></code> <code><SOUND>POLYPONIC</SOUND></code>
child::node()	Selects all child nodes of the current node	<code><COMPANY>SAMSUNG</COMPANY></code> <code><MODELNO>R220</MODELNO></code> <code><SOUND>POLYPONIC</SOUND></code>
descendant::HANDSET	Selects all HANDSET descendants of the current node	<code><COMPANY>SAMSUNG</COMPANY></code> <code><MODELNO>R220</MODELNO></code> <code><SOUND>POLYPONIC</SOUND></code>
ancestor::HANDSET	Selects all HANDSET ancestors of the current node	<code><HANDSETS></code> <code></HANDSETS></code>
ancestor-or-self::HANDSET	Selects all HANDSET ancestors of the current node - and the current as well if it is a HANDSET node	<code><HANDSET type = "GSM"></code> <code><COMPANY>NOKIA</COMPANY></code> <code><MODELNO>N91</MODELNO></code> <code><SOUND>MP3</SOUND></code> <code></HANDSET></code>
child::* / child::MODELNO	Selects all MODELNO grandchildren of the current node	<code><MODELNO>C170</MODELNO></code> <code><MODELNO>N91</MODELNO></code> <code><MODELNO>K300</MODELNO></code> <code><MODELNO>R220</MODELNO></code>



Context :

- How to use XPath expressions within an XSLT to retrieve data.



Check List :

- ☒ To understand the XPath syntax and Expressions
- ☒ To understand operators within XPath
- ☒ To understand the concept of nodes with respect to XML document and XPath



Common Errors :

- While using XPath we must concentrate on syntax as well as path mentioned.



Lessons Learnt :

- How to use XPath expression to parse an XML document and select values of various nodes



Best Practices :

- Use XPath within XSLT and understand how to select various nodes.



Topic: XQuery

Estimated Time: 120 mins.



Objectives : This module will familiarize the participant with

- Introduction to XQuery
- Use of XQuery.
- Example Of XQuery



Presentation :

- XQuery is *the* language for querying XML data: XQuery is a language for finding and extracting elements and attributes from XML documents.
- XQuery for XML is like SQL for databases.
- XQuery is built on XPath expressions: XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators. If you have already studied XPath you will have no problems understanding XQuery.
- XQuery is supported by all the major database engines (IBM, Oracle, Microsoft, etc.).
- XQuery is a W3C Recommendation

Use of XQuery:

- Extract information to use in a Web Service
- Generate summary reports
- Transform XML data to XHTML
- Search Web documents for relevant information



Scenario :

Ram wants to buy 2 books from Manilal Book Store. He places an order describing the book details. The details can be written in an XML document, order.xml, which describes the title and price of the books. XQuery is used to find out the different nodes of the document.



Demonstration/Code Snippet :

Step 1: Create a new XML document order.xml using visual studio 2005.

order.xml:

```
<?xml version="1.0" encoding="utf-8" ?>
  <order>
    <book ISBN='10-861003-324'>
      <title>The Handmaid's Tale</title>
      <price>19.95</price>
    </book>
    <cd ISBN='2-3631-4'>
      <title>Americana</title>
      <price>16.95</price>
    </cd>
  </order>
```



Knowledge of XPath: You should have knowledge of XPath to understand XQuery.

Step 2: XQuery Code to run the order.xml file

```
Imports System
Imports System.IO
Imports System.Xml
Imports System.Xml.XPath
Imports System.Windows.Forms
Public Class XQueryExample

    Private Sub XQueryExample_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Create and populate the document
        Dim doc As XPathDocument = New XPathDocument("C:\Documents and Settings"
& _
        "\bt72093\My Documents\order.xml")
        'Create the navigator
        Dim nav As XPathNavigator = doc.CreateNavigator()
        'Calculate the total of the order. Cast the result
        'to a double.
        Dim expr As XPathExpression =
nav.Compile("sum(//price/text())")
        Dim total As Double = CType(nav.Evaluate(expr), Double)
        'If the total is more than 30 dollars, give the
        'user a 5 dollar discount.
        If total > 30.0 Then
            Dim disc As Double = 5.0
            total = total - disc
        End If
    End Sub
End Class
```

```
MessageBox.Show("Total price: " & total)
Else
    MessageBox.Show("Total price: " & total)
End If
End Sub
End Class
```

Step 3: Output comes here in a message box which shows the price,

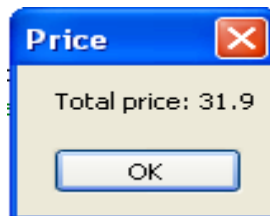


Figure 5.1-1: Output



Context :

- XQuery and its use.



Practice Session :

- Create a XML document for student data and find result of the students who scored above 60% using XQuery.



Check List :

- XQuery.
- Use of XQuery.
- Example of XQuery.



Common Errors :

- Syntax not properly defined.

- Use of doc () function not in a proper way.
- Wrong XPath.



Exceptions :



Lessons Learnt :

- ☒ Introduction to XQuery.
- ☒ Use of XQuery.
- ☒ Example of XQuery.

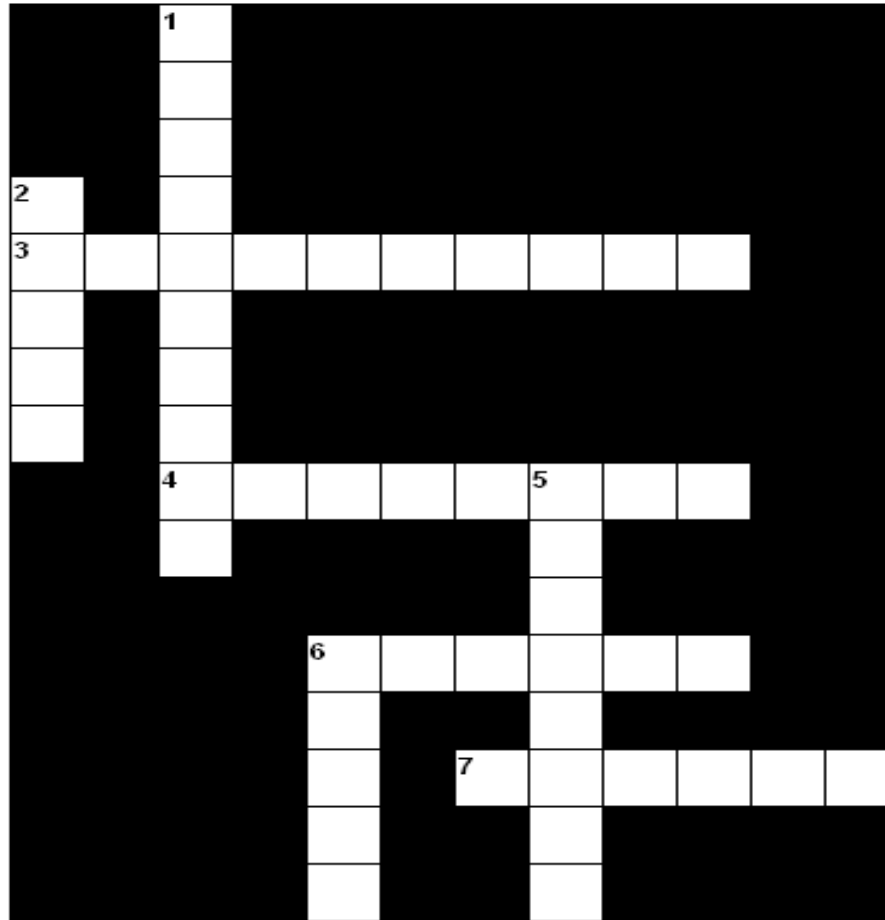


Best Practices :

- Use XQuery to retrieve data from the XML document.

Crossword Unit-5

Time: 20 minutes



Across:

3. _____ are always embedded in square brackets.
4. XPath uses path expressions to _____ in XML documents.
6. _____ is the language for querying XML data.
7. _____ values are values with no child or parent.

Down:

1. A _____ may be a node's children, children's children, etc.
2. _____ is a syntax for defining parts of an XML document.
5. An _____ may be a node's parent, parent's parent, etc.
6. Xquery transforms XML data to _____.

Answers For Crosswords

Unit-1

Across	1) tree 4) Attributes 6) xml 7) W3C
--------	-------------------------------------

Down	2) Root 3)xmlns attribute 5)SGML and HTML
------	---

Unit-2

Across	6) XSD 7) SOX
Down	1) CDATA 2) Root 3) DTD 4) XSD 5) Complex

Unit-3

Across	1) Validating 5) DOM 6) Parser 7) Larger 8) Empty
Down	2) Interface 3) Two 4) Left

Unit-4

Across	1) Transformation 3) Parent 4) Templating 5) XSLT 7) Include
Down	2) strip-space 6) Key

Unit-5

Across	3) Predicates 4) nodesset 6) Xpath 7) Atomic
Down	1) Descendants 2) Xpath 5) Ancestors 6) XHTML

Team Members



Srikanth Nivarthi
47275



Sreenivas Ram
66223



Seshu Babu Barma
56150

Veerendra Kumar Ankem
77964

Teena Arora
74572

Contributors



Bhrahmagna Trivedi
72093



Priyank Thakkar
72094



A.Shakti Patro
72073



S.Sunita Kumari
72233