

Bachelor Of Engineering In Information Technology

Semester Six, Third Year(Even semester)
30th March 2022
Offline 6th Lecture

Padre Conceicao College of Engineering
Verna Goa 403722 India

Web Technology

RC 2019-20

Unit 1

UNIT 1

Topic	Subtopics
Introduction to Web	Web Architecture, Web Applications, Web servers, Web Browsers, Overview of HTTP
HTML	Elements, Attributes, Tags, Forms, Frames, Tables, Overview and features of HTML5
Cascading Style Sheets	Need for CSS, basic syntax and structure of CSS, using CSS, background images, colors and properties, manipulating texts, using fonts, borders and boxes, margins, padding lists, positioning using CSS, Overview and features of CSS3.
XML	Introduction to XML, uses of XML, XML key components, DTD and Schemas, Transforming XML using XSL and XSL

UNIT 1 :XML

Refer Web Technologies: HTML, Javascript, PHP, Java, JSP, ASP.NET, XML and AJAX, Black Book; Publisher: Dreamtech Press(2015) ; ISBN: 978-81-7722-997-4

Sr.No	Title	Chapter	Page No.
1	Introduction to XML	4	370-389
2	uses of XML	16	413,436,439
3	XML key components	9	546,557,,561,
4	DTD and Schemas	6	
5	Transforming XML using XSL and XSL	8	

Introduction to XML

- XML stands for EXtensible Markup Language.
- XML is a meta-language, which can be used to store data & act as a mechanism to transfer information between dissimilar systems.
- XML is a markup language much like HTML.
- XML was designed to describe data.
- XML tags are not predefined in XML. You must define your own tags.
- XML is self describing.
- XML uses a DTD (Document Type Definition) to formally describe the data.

Example of an XML file

- `<?xml version="1.0"?>`
- `<Person>`
- `<Firstname>Ralph</Firstname>`
- `<Lastname>Mosely</Lastname>`
- `</Person>`

Difference between XML and HTML

Sr.No	XML	HTML
1	XML stands for e X tensible M arkup L anguage.	HTML stands for H yper T ext M arkup L anguage.
2	XML was designed to store and transfer data.	HTML was designed to display data.
3	XML focuses on what data is.	HTML focus on how data looks.
4	XML is case sensitive.	HTML is not case sensitive.
5	XML is dynamic	HTML is static
6	In XML you must design your own tag.	HTML has predefined tags.
7	XML preserves white space	HTML has no rule regarding white space
8	XML makes it mandatory to use a closing tag.	In HTML, it is optional to use a closing tag.
9	XML uses parser to check & read xml fileseg. DOM,SAX	HTML don't use any kind of parser .

Uses of XML

- **Data Transfer** (Used to exchange data between dissimilar systems.)
- **Formatting documents**(PDF files, PostScript files, Microsoft Word documents, PowerPoint documents, and RTF text files are also stored as XML.)
- **Web searching**(Search engines use HTML (XML) tags to make searches more accurate.)
- **Creating layouts**(Every layout in an Android mobile application is created in XML)
- **Storing configuration data**(Android apps not only use XML for layouts but also to store the colors, styles, and dimensions that the app will use.)
- XML is often used in front-end web development
- XML is also used in back-end web development

Some Uses of XML

- XML is also used in the following :

1. Banking services
2. Online retail stores
3. Integrating industrial systems
4. Stocks and Shares
5. Financial transactions
6. Medical data
7. Mathematical data
8. Scientific measurements
9. News information
10. Weather services

Features of XML

- XML has its own tag so it's self describing.
- Language Independent: Any language is able to read & write XML.
- OS Independent: can be work on any platform.
- Readability: It's a plain text file in user readable format so you can edit or view in simple editor.
- Hierarchical: It has hierarchical structure which is powerful to express complex data and simple to understand.

XML key components

1. XML Root Element
2. XML Element
3. XML Attribute
4. XML Namespace

1.) XML Root Element

- XML must have root element. The first element after xml version declaration in file is a root element.
- `<bookstore>` *<!-- bookstore is root element-->*
- `<book category="CHILDREN">`
- `<title>Harry Potters</title>` *<!-- <title> is an element-->*
- `<author>J K. Rowling</author>` *<!-- <author> is an element-->*
- `<year>2005</year>` *<!-- <year> is an element-->*
- `<price>99</price>` *<!-- <price> is an element-->*
- `</book>`
- `</bookstore>`
- In above example `<bookstore>` is root element.

2.) XML Element

- An XML element is everything from (including) the element's start tag to (including) the element's end tag.
- An element can contain:
 - other elements
 - text
 - attributes
 - or a mix of all of the above...
- In above example `<title>`, `<author>`, `<year>` and `<price>` are elements.

3.) XML Attribute

- Attributes provide additional information about an element.
- Attributes often provide information that is not a part of the data. In the example below, the file type is irrelevant to the data, but can be important to the software that wants to manipulate the element

<file type—"gif">computer.gif</file>

- XML Attributes Must be Quoted
- Attribute values must always be quoted. Either single or double quotes can be used. For a person's sex, the person element can be written like this:

<person height—"tall"> or <person height—"short">

4.) XML Namespace

- The XML namespace is a special type of reserved XML attribute that you place in an XML tag.
- The reserved attribute is actually more like a prefix that you attach to any namespace you create.
- This attribute prefix is "xmlns:" which stands for XML Namespace.
- The colon is used to separate the prefix from your namespace that you are creating.
- xmlns must have a unique value that no other namespace in the document has. What is commonly used is the URI (Uniform Resource Identifier) or the more commonly used URL.
- [Xmlns="http://www.mydomian.com/ns/animals/1.1"](http://www.mydomian.com/ns/animals/1.1)

How to create an XML file in Notepad

1.XML uses tags that are not predefined or standard, which means that they are created by the person who is writing the XML file.

2.Usually, the first tag begins by specifying the XML version and the encoding being used. This is a standard tag and is called the XML Prolog and looks like below:

```
<?xml version="1.0" encoding="UTF-8"?>
```

3.Encoding is needed by the browser to open the documents properly.

4.The Prolog is not mandatory but should appear as the first tag if used.

- The XML prolog is optional.
- If the XML prolog is written, it must come first in the document.
- XML documents can contain international characters, like Norwegian øæå or French êèé.
- To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.
- UTF stands for UCS Transformation Format, and UCS means Universal Character Set.
- The number 8 or 16 refers to the number of bits used to represent a character.
- They are either 8(1 to 4 bytes) or 16(2 or 4 bytes).
- For the documents without encoding information, UTF-8 is set by default.

5. Every tag used should always have a closing tag, **for example,**

<colours> </colours>

6. The tags are case sensitive. So we treat the two tags below as different tags.

<colours> <Colours>

7. Within the prolog tag are the elements which further have sub-elements within them

8. The structure is generally as below:

<root tag>

<child tag>

<sub child tag> </sub child tag>

</child tag>

</root tag>

- As seen, each of the tags should be nested in proper order.
- Those were the basic rules for creating an XML file.
- Once you keep them in mind, you can use a text editor such as Notepad to create XML files.

How to save an XML file in Notepad

- Open Notepad on your computer
- Create an XML file
- Save the file with a .xml extension

How to open a file in XML format using a web browser

- Open File Explorer and browse to the XML file that needs to be opened.
- Right-click over the file and select **Open With** to choose a web browser to open the XML file.
- The Web browser may or may not appear in the list of options.
- In case, it is not available on the list, select Choose another app.
- Now, from the displayed lists, click on **More Apps**.
- Some more options are displayed in the list. Now scroll down and select the browser in which you wish to open the file. **You can select any browser like Chrome or Internet Explorer from the list** as shown below. Select Internet Explorer and then click **OK**.
- The file opens in Internet Explorer as shown below.

How does XML file appear a web browser

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<colours>  
  <colour1>Red</colour1>  
  <colour2>Green</colour2>  
  <colour3>Blue</colour3>  
  <colour4>Yellow</colour4>  
</colours>
```

Create a XML file that contains Book Information.

book.xml

```
<?xml version="1.0"?>
<bookstore>
  <book>
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2022</year>
    <price>1020</price>
  </book>
  <book>
    <title>WAD</title>
    <author>RaIph Mosely</author>
    <year>2021</year>
    <price>513</price>
  </book>
</bookstore>
```

How does book.xml file appear in a web browser

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<bookstore>
  ▼<book>
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2022</year>
    <price>1020</price>
  </book>
  ▼<book>
    <title>WAD</title>
    <author>RaIph Mosely</author>
    <year>2021</year>
    <price>513</price>
  </book>
</bookstore>
```


Bachelor Of Engineering In Information Technology

Semester Six, Third Year(Even semester)

04th April 2022

Offline 7th Lecture

Padre Conceicao College of Engineering

Verna Goa 403722 India

DTD

- The purpose of a DTD(Document Type Definition) is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.
 - DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language.
 - A DTD can be declared inline in your XML document, or as an external reference.
1. Internal DTD
 2. External DTD

Internal DTD

- A DTD is referred to as an internal DTD if elements are declared within the XML files.
- To refer to it as internal DTD, *standalone* attribute in XML declaration must be set to **yes**. This means, the declaration works independent of an external source.
- Following is the syntax of internal DTD :

<!DOCTYPE *root-element* [*Element-Declarations*]>

where *root-element* is the name of root element and *element-declarations* is where you declare the elements.

Internal DTD

- This is an XML document with a Document Type Definition:

```
amyNote.xml - Notepad
File Edit Format View Help
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>

<note>
  <to>Amy</to>
  <from>Jim</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

How does amyNote.xml with Internal DTD file appear in a web browser

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0"?>
▼<note>
  <to>Amy</to>
  <from>Jim</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Basic Syntax of DTD

<!DOCTYPE **element** **DTD identifier**[Declarations]>

- The **DTD** starts with <!DOCTYPE delimiter.
- An **element** tells the parser to parse the document from the specified root element.
- **DTD identifier** is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called **External Subset**.
- **The square brackets []** enclose an optional list of entity declarations called *Internal Subset*.

Internal DTD for amyNote.xml

- !DOCTYPE note - Defines that the root element of the document is note
- !ELEMENT note - Defines that the note element must contain the elements: "to, from, heading, body"
- !ELEMENT to - Defines the to element to be of type "#PCDATA"
- !ELEMENT from - Defines the from element to be of type "#PCDATA"
- !ELEMENT heading - Defines the heading element to be of type "#PCDATA"
- !ELEMENT body - Defines the body element to be of type "#PCDATA"

CompanyDetail.xml with Internal DTD and Output

companyDetail.xml - Notepad

File Edit Format View Help

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
```

```
<!DOCTYPE address [
```

```
  <!ELEMENT address (name,company,phone)>
```

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

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

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

```
<address>
```

```
  <name>Webby</name>
```

```
  <company>Web Tech Pvt.Ltd</company>
```

```
  <phone>(0832) 123-4567</phone>
```

```
</address>
```

This XML file does not appear to have any style information associated with it. The document tree is shown below.

▼ <address>


<name>Webby</name>

<company>Web Tech Pvt.Ltd</company>

<phone>(0832) 123-4567</phone>

</address>

External DTD

 amyNoteExternal.xml - Notepad

File Edit Format View Help

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "amyNote.dtd">

<note>
<to>Amy</to>
<from>Jim</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

 amyNote.dtd - Notepad

File Edit Format View Help

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

Output for external DTD amyNote.dtd

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0"?>
▼<note>
  <to>Amy</to>
  <from>Jim</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

DTD Elements

- In the DTD, XML elements are declared with an element declaration. An element declaration has the following syntax:

<!ELEMENT element-name (element-content)>

Empty elements

- Empty elements are declared with the keyword EMPTY inside the parentheses:
- *<!ELEMENT img (EMPTY)>*

Elements with data

<!ELEMENT element-name (#CDATA)>

or

<!ELEMENT element-name (#PCDATA)>

or

<!ELEMENT element-name (ANY)>

example:

<!ELEMENT note (#PCDATA)>

- #CDATA means the element contains character data that is not supposed to be parsed by a parser. **XML** parsers are used to parse all the text in an **XML** document.
- #PCDATA means that the element contains data that IS going to be parsed by a parser. **PCDATA** stands for Parsed Character data. If a #PCDATA section contains elements, these elements must also be declared.
- The keyword ANY declares an element with any content.

Elements with children (sequences)

- Elements with one or more children are defined with the name of the children elements inside the parentheses:

<!ELEMENT element-name (child-element-name)>

Or

<!ELEMENT element-name (child-element-name,child-element-name,..)>

Example:

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

Elements with children (sequences)

- When children are declared in a sequence separated by commas, the children must appear in the same sequence in the document. In a full declaration, the children must also be declared, and the children can also have children. The full declaration of the note document will be:

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

<!ELEMENT to (#CDATA)>

<!ELEMENT from (#CDATA)>

<!ELEMENT heading (#CDATA)>

<!ELEMENT body (#CDATA)>

Wrapping

- If the DTD is to be included in your XML source file, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element [Element-Declarations]>
```

Refer to amyNote.xml

Declaring only one occurrence of the same element

- The example declaration below declares that the child element message can only occur one time inside the note element.

<!ELEMENT element-name (child-name)>

example

<!ELEMENT note (message)>

Declaring minimum one occurrence of the same element

<!ELEMENT element-name (child-name+)>

example

<!ELEMENT note (message+)>

- The **+** sign in the example above declares that the child element message must occur one or more times inside the note element.

Declaring zero or more occurrences of the same element

<!ELEMENT element-name (child-name)>*

example

<!ELEMENT note (message)>*

- The * sign in the example above declares that the child element message can occur zero or more times inside the note element.

Declaring zero or one occurrences of the same element

<!ELEMENT element-name (child-name?)>

example

<!ELEMENT note (message?)>

The **?** Sign in the example above declares that the child element message can occur zero or one time inside the note element.

Bachelor Of Engineering In Information Technology

Semester Six, Third Year(Even semester)

05th April 2022

Offline: 8th Lecture

Padre Conceicao College of Engineering

Verna Goa 403722 India

DTD Attributes

- Attributes provide extra information about elements.
- Attributes are placed inside the start tag of an element.

Declaring Attributes

- In the DTD, XML element attributes are declared with an ATTLIST declaration.
- An attribute declaration has the following syntax:

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

- The ATTLIST declaration defines the element which can have the attribute, the name of the attribute, the type of the attribute, and the default attribute value.
- The attribute-type can have the following values:

The **attribute-type** can be one of the following

Type	Description
CDATA	The value is character data
(<i>en1 en2 ..</i>)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is a predefined xml value

The **attribute-value** can be one of the following

Value	Explanation
<i>value</i>	The default value of the attribute
#REQUIRED	The attribute is required
#IMPLIED	The attribute is optional
#FIXED <i>value</i>	The attribute value is fixed

A Default Attribute Value

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

Valid XML:

<square width="100" />

- In the above example, the "square" element is defined to be an empty element with a "width" attribute of type CDATA.
- If no width is specified, it has a default value of 0.

#REQUIRED

- Use the #REQUIRED keyword if you don't have an option for a default value, but still want to force the attribute to be present.
- Syntax

```
<!ATTLIST element-name attribute-name attribute-type  
#REQUIRED>
```

Example

DTD:

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

Valid XML:

```
<person number="5677" />
```

Invalid XML:

```
<person />
```

#IMPLIED

- Use the #IMPLIED keyword if you don't want to force the author to include an attribute, and you don't have an option for a default value.

- Syntax

<!ATTLIST element-name attribute-name attribute-type #IMPLIED>

- Example

- DTD:

<!ATTLIST contact fax CDATA #IMPLIED>

Valid XML:

<contact fax="123-456789" />

Invalid XML:

<contact />

#FIXED

- Use the #FIXED keyword when you want an attribute to have a fixed value without allowing the author to change it. If an author includes another value, the XML parser will return an error.

- Syntax

`<!ATTLIST element-name attribute-name attribute-type #FIXED "value">`

- Example

- DTD:

`<!ATTLIST sender company CDATA #FIXED "Microsoft">`

Valid XML:

`<sender company="Microsoft" />`

Invalid XML:

`<sender company="WebTech" />`

XML Schemas

- An XML Schema describes the structure of an XML document.
- XML Schema is an XML-based alternative to DTD.
- The XML Schema language is also referred to as XML Schema Definition (XSD).
- XML Schema is a W3C Recommendation.

XML Schemas Support Data Types

- One of the greatest strength of XML Schemas is the support for data types.
- It is easier to describe allowable document content
- It is easier to validate the correctness of data
- It is easier to define data facets (restrictions on data)
- It is easier to define data patterns (data formats)
- It is easier to convert data between different data types

XSD Elements

- XML Schemas define the elements of your XML files. It's of two types:
 - 1. Simple**
 - 2. Complex Type**

XSD Simple Elements

- A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.
- The "only text" restriction is misleading.
- The text can be of many different types. It can be one of the types included in the XML Schema definition (boolean, string, date, etc.), or it can be a custom type that you can define yourself.
- You can also add restrictions (facets) to a data type in order to limit its content, or you can require the data to match a specific pattern.

Defining a Simple Element

The syntax for defining a simple element is:

```
<xs : element name="nnn" type="ttt"/>
```

- where **nnn** is the name of the element and **ttt** is the data type of the element.
- XML Schema has a lot of built-in data types. The most common types are:
 1. xs:string
 2. xs:decimal
 3. xs:integer
 4. xs:boolean
 5. xs:date
 6. xs:time

Example

- Below are some XML elements:

```
<lastname>will</lastname>
```

```
<age>1</age>
```

```
<dateborn>2020-04-05</dateborn>
```

Below are the corresponding simple element definitions:

```
<xs:element name="lastname" type="xs:string"/>
```

```
<xs:element name="age" type="xs:integer"/>
```

```
<xs:element name="dateborn" type="xs:date"/>
```

XSD Complex Elements

- A complex element is an XML element that contains other elements and/or attributes.
- There are four kinds of complex elements:
 1. empty elements
 2. elements that contain only other elements
 3. elements that contain only text
 4. elements that contain both other elements and text

Examples of Complex Elements

- A complex XML element, "package", which is empty:

```
<package pid="1241"/>
```

- A complex XML element, "student", which contains only other elements:

```
<student>  
  <firstname>Ally</firstname>  
  <lastname>West</lastname>  
</student>
```

- A complex XML element, "milkshake", which contains only text:

```
<milkshake type="dessert">chocolate with Ice cream</milkshake>
```

- A complex XML element, "description", which contains both elements and text:

```
<description>  
The event took place on <date lang="norwegian">01.04.22</date> ....  
</description>
```

XSD Attributes

- Simple elements cannot have attributes. If an element has attributes, it is considered to be of a complex type. But the attribute itself is always declared as a simple type.

How to Define an XSD Attribute?

- The syntax for defining an attribute is:

```
<xs:attribute name="www" type="vvv"/>
```

where **www** is the name of the attribute and **vvv** specifies the data type of the attribute.

Default and Fixed Values for Attributes

- Attributes may have a default value OR a fixed value specified.
- A default value is automatically assigned to the attribute when no other value is specified.
- In the following example the default value is "EN":

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

- A fixed value is also automatically assigned to the attribute, and you cannot specify another value.
- In the following example the fixed value is "EN":

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

Optional and Required Attributes

- Attributes are optional by default. To specify that the attribute is required, use the "use" attribute:

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```


Bachelor Of Engineering In Information Technology

Semester Six, Third Year(Even semester)

06th April 2022

Offline: 9th Lecture

Padre Conceicao College of Engineering

Verna Goa 403722 India

Difference between DTD and XML schema

	DTD	XML Schema
Markup Validation	Can specify only the root element in the instance document	Any global element can be the root
	No ambiguous content support	No ambiguous content support
Namespace support	No	Yes. Declarations only where multiple namespaces are used
Code reuse	Poorly supported. Can use parameter entities.	Can reuse definitions using named types.
Data Type validation	No real data type support	Provides flexible set of data types.

What is XSL?

- XSL stands for EXtensible Stylesheet Language.
- XSL = Style Sheets for XML
- XSL describes how the XML document should be displayed
- XSL- More Than a Style Sheet Language
- XSL consists of three parts:
 1. XSLT-a language for transforming XML documents
 2. XPath-a language for navigating in XML documents
 3. XSL-FO-a language for formatting XML documents

What is XSLT?

- XSLT stands for XSL Transformations.
- XSLT is the most important part of XSL.
- XSLT transforms an XML document into another XML document.
- XSLT uses XPath to navigate in XML documents.

XPath expression

- An XPath expression generally defines a pattern in order to select a set of nodes. These patterns are used by XSLT to perform transformations or by XPointer for addressing purpose.
- XPath specification specifies seven types of nodes which can be the output of execution of the XPath expression.
 1. Root
 2. Element
 3. Text
 4. Attribute
 5. Comment
 6. Processing Instruction
 7. Namespace
- XPath uses a path expression to select node or a list of nodes from an XML document.

list of useful paths and expression to select any node/ list of nodes from an XML document.

Sr.No	Expression	Description
1	node-name	Select all nodes with the given name "nodename"
2	/	Selection starts from the root node
3	//	Selection starts from the current node that match the selection
4	.	Selects the current node
5	..	Selects the parent of the current node
6	@	Selects attributes
7	student	Example – Selects all nodes with the name "student"
8	class/ student	Example – Selects all student elements that are children of class
9	//student	Selects all student elements no matter where they are in the document

Explain XSL Transformation and XSL Elements

- The style sheet provides the template that transforms the document from one structure to another.
- In this case `<xsl:template>` starts the definition of the actual template, as the root of the source XML document.
- The `match = "/"` attribute makes sure this begins applying the template to the root of the source XML document.

How to link XSL to XML

- The style sheet is linked into the XML by adding the connecting statement to the XML document:

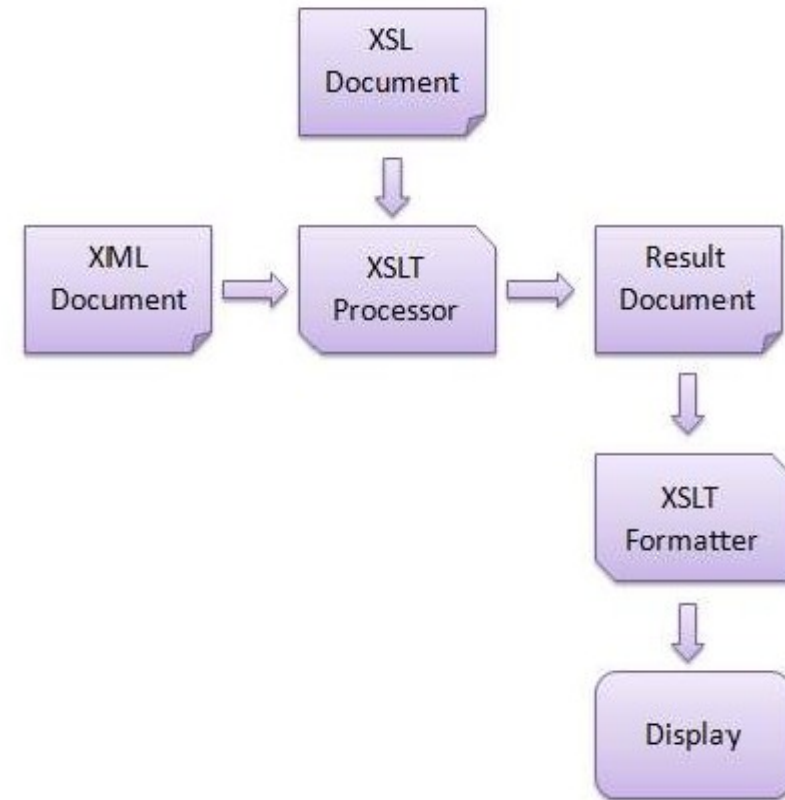
```
<?xml-stylesheet type="text/xsl" href="libstyle.xsl" ?>
```


XSL Transformations

- XSLT is the most important part of XSL.
- XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.
- With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.
- A common way to describe the transformation process is to say that XSLT transforms an XML source-tree into an XML result-tree.

How XSLT Works

- An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.



- XSLT Uses XPath:
 - XSLT uses XPath to find information in an XML document.
 - XPath is used to navigate through elements and attributes in XML documents.
- XSLT Works as:
 - In the transformation process, XSLT uses XPath to define parts of the source document that should match one or more predefined templates.
 - When a match is found, XSLT will transform the matching part of the source document into the result document.

XSL Elements

- XSL contains many elements that can be used to manipulate, iterate and select XML, for output.
 1. value-of
 2. for-each
 3. sort
 4. if
 5. choose

<xsl:value-of> Element

- The `<xsl:value-of>` element extracts the value of a selected node.
- The `<xsl:value-of>` element can be used to select the value of an XML element and add it to the output.
- Syntax

`<xsl:value-of select="expression" />`

- expression: This is Required.
- An XPath expression that specifies which node/attribute to extract the value from.
- It works like navigating a file system where a forward slash (/) selects subdirectories.

syntax declaration of **<xsl:value-of>** element

```
<xsl:value-of  
    Select=Expression  
    Disable-output="yes"|"no">  
</xsl:value-of>
```

Attributes of <xsl:value-of> element

Sr.No	Name	Description
1	Select	XPath Expression to be evaluated in current context.
2	disable-outputescaping	Default-"no". If "yes", output text will not escape xml characters from text.

Example of **<xsl:value-of>** element

- This example creates a table of **<student>** element with its attribute **rollno** and its child **<firstname>**, **<lastname>**, **<nickname>**, and **<marks>**

```
*students.xml - Notepad
File Edit Format View Help
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "student.xml"?>
<class>
  <student rollno = "07">
    <firstname>Blase</firstname>
    <lastname>Vaz</lastname>
    <nickname>Nash</nickname>
    <marks>90</marks>
  </student>
  <student rollno = "11">
    <firstname>Abhiraj</firstname>
    <lastname>Dessai</lastname>
    <nickname>Baburao</nickname>
    <marks>95</marks>
  </student>
  <student rollno = "36">
    <firstname>Simran</firstname>
    <lastname>Vaz</lastname>
    <nickname>Ventura</nickname>
    <marks>85</marks>
  </student>
</class>
```

```
student.xml - Notepad
File Edit Format View Help
<?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>Students</h2>
        <table border = "1">
          <tr bgcolor = "#9acd32">
            <th>Roll No</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Nick Name</th>
            <th>Marks</th>
          </tr>
          <xsl:for-each select = "class/student">
            <tr>
              <td><xsl:value-of select = "@rollno"/></td>
              <td><xsl:value-of select = "firstname"/></td>
              <td><xsl:value-of select = "lastname"/></td>
              <td><xsl:value-of select = "nickname"/></td>
              <td><xsl:value-of select = "marks"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```


Output for Example of **<xsl:value-of>** element

Students

Roll No	First Name	Last Name	Nick Name	Marks
07	Blase	Vaz	Nash	90
11	Abhiraj	Dessai	Baburao	95
36	Simran	Vaz	Ventura	85

<xsl:for-each> Element

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

```
<xsl:for-each
```

```
    select=Expression>
```

```
</xsl:for-each>
```

Attributes of **<xsl:for-each>** element

Sr.No	Name	Description
1	Select	XPath Expression to be evaluated in current context to determine the set of nodes to be iterated

Example of **<xsl:for-each>** element

- This example creates a table of **<student>** element with its attribute **rollno** and its child **<firstname>**, **<lastname>**, **<nickname>**, and **<marks>** by iterating over

```
*students.xml - Notepad
File Edit Format View Help
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "student.xsl"?>
<class>
  <student rollno = "07">
    <firstname>Blase</firstname>
    <lastname>Vaz</lastname>
    <nickname>Nash</nickname>
    <marks>90</marks>
  </student>
  <student rollno = "11">
    <firstname>Abhiraj</firstname>
    <lastname>Dessai</lastname>
    <nickname>Baburao</nickname>
    <marks>95</marks>
  </student>
  <student rollno = "36">
    <firstname>Simran</firstname>
    <lastname>Vaz</lastname>
    <nickname>Ventura</nickname>
    <marks>85</marks>
  </student>
</class>
```

```
student.xsl - Notepad
File Edit Format View Help
<?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>Students</h2>
        <table border = "1">
          <tr bgcolor = "#9acd32">
            <th>Roll No</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Nick Name</th>
            <th>Marks</th>
          </tr>
          <xsl:for-each select = "class/student">
            <tr>
              <td><xsl:value-of select = "@rollno"/></td>
              <td><xsl:value-of select = "firstname"/></td>
              <td><xsl:value-of select = "lastname"/></td>
              <td><xsl:value-of select = "nickname"/></td>
              <td><xsl:value-of select = "marks"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Output for Example of **<xsl:for-each>** element

Students

Roll No	First Name	Last Name	Nick Name	Marks
07	Blase	Vaz	Nash	90
11	Abhiraj	Dessai	Baburao	95
36	Simran	Vaz	Ventura	85

<xsl:if> Element

- To put a conditional if test against the content of the XML file, add an <xsl:if> element to the XSL document.
- Syntax

```
<xsl:if  
    test=boolean-expression>  
</xsl:if>
```

Attributes of **<xsl:if>** element

Sr.No	Name	Description
1	test	The condition in the xml data to test.

Example of **<xsl:if>** element

- This example creates a table of `<student>` element with its attribute **rollno** and its child `<firstname>`, `<lastname>`, `<nickname>`, and `<marks>` by iterating over each student.
- It checks marks to be greater than 90 and then prints the student(s) details.

Example of <xsl:if> element with output

studentif.xml - Notepad

File Edit Format View Help

```
<?xml version = "1.0"?>
```

```
<?xml-stylesheet type = "text/xsl" href = "studentif.xsl"?>
```

```
<class>
```

```
  <student rollno = "07">
```

```
    <firstname>Blase</firstname>
```

```
    <lastname>Vaz</lastname>
```

```
    <nickname>Nash</nickname>
```

```
    <marks>90</marks>
```

```
  </student>
```

```
  <student rollno = "11">
```

```
    <firstname>Abhiraj</firstname>
```

```
    <lastname>Dessai</lastname>
```

```
    <nickname>Baburao</nickname>
```

```
    <marks>95</marks>
```

```
  </student>
```

```
  <student rollno = "36">
```

```
    <firstname>Simran</firstname>
```

```
    <lastname>Vaz</lastname>
```

```
    <nickname>Ventura</nickname>
```

```
    <marks>85</marks>
```

```
  </student>
```

```
</class>
```

studentif.xsl - Notepad

File Edit Format View Help

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

```
  <html>
```

```
    <body>
```

```
      <h2>Students</h2>
```

```
      <table border = "1">
```

```
        <tr bgcolor = "#9acd32">
```

```
          <th>Roll No</th>
```

```
          <th>First Name</th>
```

```
          <th>Last Name</th>
```

```
          <th>Nick Name</th>
```

```
          <th>Marks</th>
```

```
        </tr>
```

```
        <xsl:for-each select = "class/student">
```

```
          <xsl:if test = "marks > 90">
```

```
            <tr>
```

```
              <td><xsl:value-of select = "@rollno"/></td>
```

```
              <td><xsl:value-of select = "firstname"/></td>
```

```
              <td><xsl:value-of select = "lastname"/></td>
```

```
              <td><xsl:value-of select = "nickname"/></td>
```

```
              <td><xsl:value-of select = "marks"/></td>
```

```
            </tr>
```

```
          </xsl:if>
```

```
        </xsl:for-each>
```

```
      </table>
```

```
    </body>
```

```
  </html>
```

```
</xsl:template>
```

```
</xsl:stylesheet>
```

Students

Roll No	First Name	Last Name	Nick Name	Marks
11	Abhiraj	Dessai	Baburao	95

<xsl:sort> Element

- The `<xsl:sort>` element is used to sort the output.
- Following is the syntax declaration of `<xsl:sort>` element.

```
<xsl:sort  
    select=string-expression  
    Lang={nmtoken}  
    Data-type={"text"|"number"|QName}  
    Order={"ascending"|"descending"}  
    Case-order={"upper-first"|"lower-first"}>  
</xsl:sort >
```

The select attribute indicates what XML element to sort on.

Attributes of **<xsl:sort>** element

Sr.No	Name	Description
1	select	Sorting key of the node.
2	lang	Language alphabet used to determine sort order.
3	data-type	Data type of the text.
4	order	Sorting order. Default is "ascending".
5	case-order	Sorting order of string by capitalization. Default is "upper-first".

Example of **<xsl:sort>** element with output

- This example creates a table of `<student>` element with its attribute **rollno** and its child `<firstname>`, `<lastname>`, `<nickname>`, and `<marks>` by iterating over each student sort them by marks.
- The student data with the lowest marks will be displayed first.
- The student data with the highest marks will be displayed last.

Example of `<xsl:sort>` element with output

studentsort.xml - Notepad

File Edit Format View Help

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "studentsort.xsl"?>
<class>
  <student rollno = "07">
    <firstname>Blase</firstname>
    <lastname>Vaz</lastname>
    <nickname>Nash</nickname>
    <marks>90</marks>
  </student>
  <student rollno = "11">
    <firstname>Abhiraj</firstname>
    <lastname>Dessai</lastname>
    <nickname>Baburao</nickname>
    <marks>95</marks>
  </student>
  <student rollno = "36">
    <firstname>Simran</firstname>
    <lastname>Vaz</lastname>
    <nickname>Ventura</nickname>
    <marks>85</marks>
  </student>
</class>
```

studentsort.xsl - Notepad

File Edit Format View Help

```
<?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>Students</h2>
        <table border = "1">
          <tr bgcolor = "#9acd32">
            <th>Roll No</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Nick Name</th>
            <th>Marks</th>
          </tr>
          <xsl:for-each select = "class/student"> |
            <xsl:sort select = "marks"/>
            <tr>
              <td><xsl:value-of select = "@rollno"/></td>
              <td><xsl:value-of select = "firstname"/></td>
              <td><xsl:value-of select = "lastname"/></td>
              <td><xsl:value-of select = "nickname"/></td>
              <td><xsl:value-of select = "marks"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Output of example of **<xsl:sort>** element

Students

Roll No	First Name	Last Name	Nick Name	Marks
36	Simran	Vaz	Ventura	85
07	Blase	Vaz	Nash	90
11	Abhiraj	Dessai	Baburao	95

Transforming XML using XSL and XSLT

- Assume we have the following sample XML file, students.xml, which is required to be transformed into a well-formatted HTML document.

```
<?xml version = "1.0"?>
<class>
  <student rollno = "07">
    <firstname>Blase</firstname>
    <lastname>Vaz</lastname>
    <nickname>Nash</nickname>
    <marks>90</marks>
  </student>
  <student rollno = "11">
    <firstname>Abhiraj</firstname>
    <lastname>Dessai</lastname>
    <nickname>Baburao</nickname>
    <marks>95</marks>
  </student>
  <student rollno = "36">
    <firstname>Simran</firstname>
    <lastname>Vaz</lastname>
    <nickname>Ventura</nickname>
    <marks>85</marks>
  </student>
</class>
```

XSLT steps

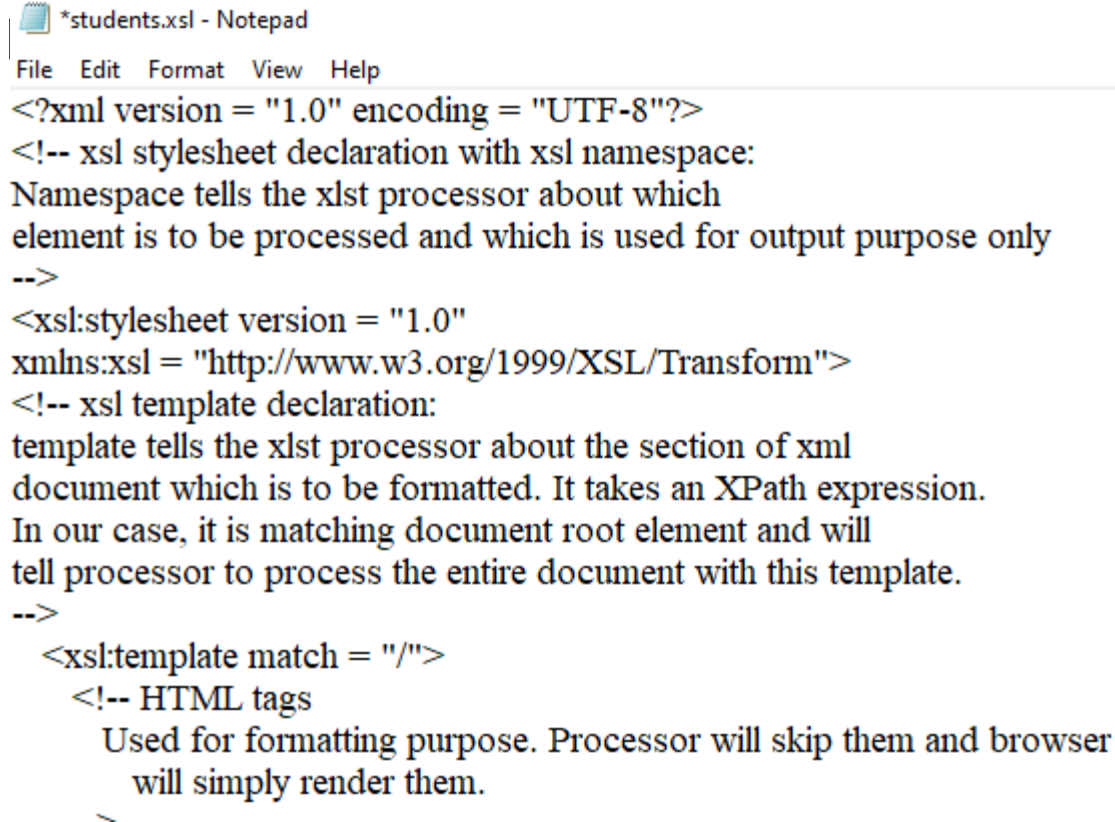
- We need to define an XSLT style sheet document for the above XML document to meet the following criteria –
- Page should have a title **Students**.
- Page should have a table of student details.
- Columns should have following headers: Roll No, First Name, Last Name, Nick Name, Marks
- Table must contain details of the students accordingly.

Advantages of using XSLT

1. Independent of programming.
2. Transformations are written in a separate xsl file which is again an XML document.
3. Output can be altered by simply modifying the transformations in xsl file.
4. No need to change any code.
5. So Web designers can edit the stylesheet and can see the change in the output quickly.

Step 1: Create XSLT document

- Use any text editor such as Notepad.
- Create an XSLT document to meet the above requirements, name it as students.xsl and save it in the same locatio



```
*students.xsl - Notepad
File Edit Format View Help
<?xml version = "1.0" encoding = "UTF-8"?>
<!-- xsl stylesheet declaration with xsl namespace:
Namespace tells the xslt processor about which
element is to be processed and which is used for output purpose only
-->
<xsl:stylesheet version = "1.0"
xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
<!-- xsl template declaration:
template tells the xslt processor about the section of xml
document which is to be formatted. It takes an XPath expression.
In our case, it is matching document root element and will
tell processor to process the entire document with this template.
-->
<xsl:template match = "/">
  <!-- HTML tags
    Used for formatting purpose. Processor will skip them and browser
    will simply render them.
  -->
```

Step 2: Link the XSLT Document to the XML Document

- Update student.xml document with the following xml-stylesheet tag. Set href value to students.xsl

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/xsl" href="students.xsl"?>
```

Updated student.xml

```
students.xml - Notepad
File Edit Format View Help
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>
<class>
  <student rollno = "07">
    <firstname>Blase</firstname>
    <lastname>Vaz</lastname>
    <nickname>Nash</nickname>
    <marks>90</marks>
  </student>
  <student rollno = "11">
    <firstname>Abhiraj</firstname>
    <lastname>Dessai</lastname>
    <nickname>Baburao</nickname>
    <marks>95</marks>
  </student>
  <student rollno = "36">
    <firstname>Simran</firstname>
    <lastname>Vaz</lastname>
    <nickname>Ventura</nickname>
    <marks>85</marks>
  </student>
</class>
```

Step 3: View the XML Document in Internet Explorer



Students

Roll No	First Name	Last Name	Nick Name	Marks
07	Blase	Vaz	Nash	90
11	Abhiraj	Dessai	Baburao	95
36	Simran	Vaz	Ventura	85

Semester questions

August 2020 Examination

Part B

- Q6b) Sketch and Explain Porter's External Agent Model(6 marks)
- Arya fig 10.2,slide 8

Part C

- Q8b) Summarize any 3 information system techniques that can be used to gain competitive advantage (12 marks)
Fiza,slide 34

Assignment 1

- Q4) Create a XML file that contains family Information (5 marks)
- *Assignment Announced to students : AA :06th April 2022*
- *Assignment to be Submitted by students : AS: 18th April 2022*