Unit II

CHAPTER 4

XML

University Prescribed Syllabus

Comparing XML with HTML, Advantages and Disadvantages of XML, Structure of an XML Document, XML Entity References, DTD, XSLT: XSLT Elements and Attributes. xsl:template, xsl:apply-templates, xsl:import, xsl:call-template, xsl:include, xsl:element, xsl:attribute, xsl:attribute-set, xsl:value-of

4.1 Introduction

☞ **Explain the role of XML in web programming.**

☞ **Write a short note on XML.**

- XML is a software- and hardware-independent tool for storing an transporting data.
- XML (eXtensible Markup Language) is a mark up language.
- XML is designed to store and transport data.
- XML was released in late 90's. it was created to provide an easy to use an store self describing data.
- XML was designed to store and transport data, it means XML is designed t carry data, not to display data.
- XML is not a replacement for HTML.
- XML is designed to be self-descriptive.

- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.
- See the example of XML file. May be it is a little hard to understand, but XML does not do anything.

```
<? Xml version="1.0" standalone="yes"?>
<BankAccount>
<Number>1234</Number>
<Type>Checking</Type>
<OpenDate>11/04/1974<OpenDate>
<Balance>25382.20</Balance>
<AccountHolder>
<LastName>Singh</LastName>
<FirstName>Darshan</FirstName>
</AccountHolder>
</BankAccount>
```

- The XML above is quite self-descriptive: It has Account Number, Type of Account, Account opening date, Current Balance, details of Account holder such as First Name, Last Name etc..
- XML is just information wrapped in tags. Someone must write a piece of software to send, receive, store, or display it.

Syllabus Topic : Comparing XML with HTML

4.2 Comparing XML with HTML

☞ **Discuss the difference between XML and HTML.**

- XML and HTML were designed with different goals.
- XML was designed to carry data - with focus on what data is?
- HTML was designed to display data - with focus on how data looks.
- XML tags are not predefined like HTML tags are.

Table 4.2.1: Comparison between HTML and XML

HTML	XML
HTML is an abbreviation for HyperText Markup Language.	XML stands for eXtensible Markup Language.
HTML was designed to display data with focus on how data looks.	XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.
HTML is a presentation language.	XML is neither a programming language nor a presentation language.
HTML is a markup language itself.	XML provides a framework for defining markup languages.
HTML does not preserve white space.	XML preserves white space.
HTML is used for designing a web-page to be rendered on the client side.	XML is used basically to transport data between the application and the database.
HTML is not strict if the user does not use the closing tags.	XML makes it mandatory for the user the close each tag that has been used.
HTML is case insensitive.	XML is case sensitive.
HTML has it own predefined tags.	While what makes XML flexible is that custom tags can be defined and the tags are invented by the author of the XML document.
HTML is about displaying data,hence static.	XML is about carrying information,hence dynamic.

Syllabus Topic : Advantages of XML

4.2.1 Advantages of XML

- Explain the advantages of XML.
- Discuss the benefits of XML over HTML.

- XML is widely used in the era of web development. It is also used to simplify data storage and data sharing.
- The main features or advantages of XML are given in Fig. 4.2.1.

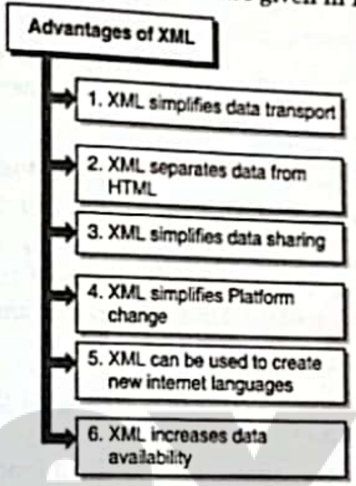

Advantages of XML
1. XML simplifies data transport
2. XML separates data from HTML
3. XML simplifies data sharing
4. XML simplifies Platform change
5. XML can be used to create new internet languages
6. XML increases data availability

Fig. 4.2.1 : Advantages of XML.

(1) XML simplifies data transport

- One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet.
- Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

(2) XML separates data from HTML

- If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes.
- With XML, data can be stored in separate XML files. This way you can focus on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTML.
- With a few lines of JavaScript code, you can read an external XML file and update the data content of your web page.

(3) XML simplifies data sharing

- XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data.
- This makes it much easier to create data that can be shared by different applications.

(4) XML simplifies Platform change

- Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost.
- XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

(5) XML can be used to create new internet languages

- A lot of new Internet languages are created with XML.
- Here are some examples: XHTML, WSDL for describing available web services, WAP and WML as markup languages for handheld devices, RSS languages for news feeds, SMIL for describing multimedia for the web.

(6) XML increases data availability

- Different applications can access your data, not only in HTML pages, but also from XML data sources.
- With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.

Syllabus Topic : Disadvantages of XML

4.2.2 Disadvantages of XML

☞ **Discuss the disadvantages of XML.**

The main disadvantages of XML are given in Fig. 4.2.2.

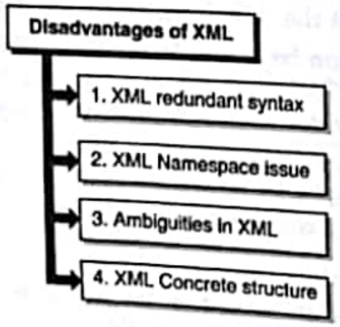


Fig. 4.2.2

(1) XML redundant syntax

- XML syntax is redundant or large relative to binary representations of similar data especially with tabular data.
- The redundancy may affect application efficiency through higher storage, transmission and processing costs Expressing overlapping (non-hierarchical) node relationships requires extra effort.

(2) XML Namespace issue

XML namespaces are problematic to use and namespace support can be difficult to correctly implement in an XML parser.

(3) Ambiguities in XML

- XML is commonly depicted as "self-documenting" but this depiction ignores critical ambiguities. The redundancy may affect application efficiency through higher storage, transmission and processing.
- The distinction between content and attributes in XML seems unnatural to some and makes designing XML data structures harder.
- Transformations, even identity transforms, result in changes to format (whitespace, attribute ordering, attribute quoting, whitespace around attributes, newlines). These problems can make diff-ing the XML source very difficult.

(4) XML Concrete structure

- The hierarchical model for representation is limited in comparison to an object oriented graph XML syntax is verbose, especially for human readers, relative to other alternative 'text-based' data transmission formats.
- Encourages non-relational data structures (data non-normalized)
- XML is very concrete and highly non-canonical. It introduces a very strong coupling between the actual representation chosen and the processing program (unlike relational storage and SQL).

Syllabus Topic : Structure of an XML Document

4.3 Structure of an XML Document

☞ **Explain the structure of XML Document.**

Discuss the structure of XML Document with suitable example.

- XML documents form a tree structure that starts at "the root" and branches to "the leaves". XML documents are formed as element trees.
- An XML tree starts at a root element and branches from the root to child elements. All elements can have sub elements (child elements):

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

- The terms parent, child, and sibling are used to describe the relationships between elements.
- Parent have children. Children have parents. Siblings are children on the same level (brothers and sisters). All elements can have text content and attributes category.

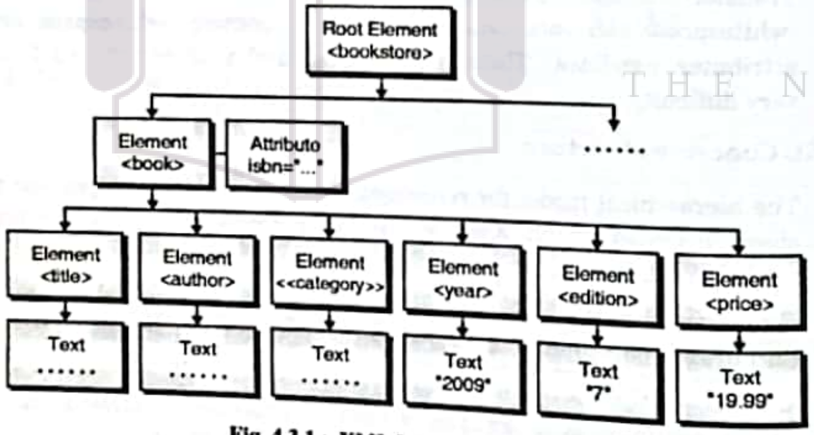


Fig. 4.3.1 : XML Document Structure

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<stud_info>
<stud category="Science">
```

```
<snamelang="en">Rohit Sharma</sname>
<rno>459</rno>
<year>2017</year>
<marks>890</marks>
</stud>

<stud category="Arts">
<snamelang="en">Jessica Smith</sname>
<rno>222</rno>
<year>2015</year>
<marks>867</marks>
</stud>

<stud category="Commerce">
<snamelang="en">RuchiDoshi</sname>
<rno>178</rno>
<year>2017</year>
<marks>933</marks>
</stud>
</stud_info>
```

- XML uses a much self-describing syntax.A prolog defines the XML version and the character encoding:<?xml version="1.0" encoding="UTF-8"?>

  The next line is the root element of the document :

```
<stud_info>
```

- The next line starts a <stud> element, which stores student's detailed information. **<stud category="Science">**. The <stud> elements have 4 child elements: <sname>, <rno>, <year>, <marks>.

  This will generate following output :

```
Rohit Sharma
459
2017
890
```

```
Jessica Smith
222
2015
867

Ruchi Doshi
178
```

Syllabus Topic : XML Entity References

4.4 XML Entity References

What do you mean by XML Entity References. Explain it in detail.

There are two additional ways in XML documents to reference characters :

1. a numeric character reference refers to a character
2. an entity reference refers to a series of characters

Where entities are used, the referenced characters appear.

4.4.1 Numeric character references

A numeric character reference can be specified in two formats :

```
&#nnnn; or &#xhhhh
```

where the n decimal digits or the h hexadecimal characters identify the Unicode character code of the referenced character.

4.4.2 Entity references

- An entity reference is an alternative name for a series of characters. You can use an entity in the &name; format, where **name** is the name of the entity. There are some predefined entities in XML, furthermore you can declare entities in a DTD (Document Type Definition).
- An entity reference is a group of characters used in text as a substitute for a single specific character that is also a markup delimiter in XML. Using the entity reference prevents a literal character from being mistaken for a markup delimiter

- For example, if an attribute must contain a left angle bracket (<), you can substitute the entity reference "&lt;". Entity references always begin with an ampersand (&) and end with a semicolon (;). You can also substitute a numeric or hexadecimal reference.

1. **Predefined entities**

The entities predefined in XML are identified in the following table.

Character	Entity reference	Numeric reference	Hexadecimal reference
&	&amp;	&#38;	&#x26;
<	&lt;	&#60;	&#x3C;
>	&gt;	&#62;	&#x3E;
"	&quot;	&#34;	&#x22;
'	&apos;	&#39;	&#x27;

2. **Declaring entities**

You can declare entities in a DTD in the following format:

```
<!ENTITY name "text">
```

where name is the name of the entity and text is the referenced text that appears where the entity is used.

Example

This example declares an entity named 'sname' and uses it and the predefined 'lt' entity in an XML document :

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE sample [
<!NOTATION vrml PUBLIC "VRML 1.0">
<!ENTITY sname "Students_Name">
]>
<entityTest>
<predefined>2 &lt; 5</predefined>
<declared>&sname;</declared>
</entityTest>
```

The document tree of the previous example :

```
<entityTest>
<predefined>2 < 5</predefined>
<declared>Students_Name</declared>
</entityTest>
```

Syllabus Topic : DTD

4.5 DTD

What is DTD? Explain how to decalre it.

Discuss the different types of DTD.

- An XML document with correct syntax is called "Well Formed". An XML document validated against a DTD is both "Well Formed" and "Valid". The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.
- The XML Document Type Declaration, commonly known as DTD, is a way to describe XML language precisely. DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language.
- n XML DTD can be either specified inside the document, or it can be kept in separate document and then liked separately.

tax

Basic syntax of a DTD is as follows :

```
<!DOCTYPE element DTD identifier
[
  declaration1
  declaration2
  ........
]>
```

In the above syntax,

- 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.

4.5.1 Internal DTD

Write a short note on Internal DTD.

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

Syntax

The syntax of internal DTD is as shown :

```
<!DOCTYPE root-element [element-declarations]>
```

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

Example

Following is a simple example of internal DTD :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE book [
<!ELEMENT book (bname,author,price)>
<!ELEMENTbname (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
]>
<book>
<bname>TanmayPatil</bname>
<author>TutorialsPoint</ author >
<price>(011) 123-4567</ price
</book>
```

Immediately after the XML header, the document type declaration follows, commonly referred to as the DOCTYPE :

```
<!DOCTYPE book[
```

The DOCTYPE declaration has an exclamation mark (!) at the start of the element name. The DOCTYPE informs the parser that a DTD is associated with this XML document.

DTD Body

- The DOCTYPE declaration is followed by body of the DTD, where you declare elements, attributes, entities, and notations:

```
<!ELEMENT book (bname,author,price)>
<!ELEMENTbname (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

- Several elements are declared here that make up the vocabulary of the <bname> document.
- <!ELEMENT bname (#PCDATA)> defines the element name to be of type "#PCDATA". Here #PCDATA means parse-able text data.
- Finally, the declaration section of the DTD is closed using a closing bracket and a closing angle bracket (]>). This effectively ends the definition, and thereafter, the XML document follows immediately.

Rules

1. The document type declaration must appear at the start of the document (preceded only by the XML header) - it is not permitted anywhere else within the document.
2. Similar to the DOCTYPE declaration, the element declarations must start with an exclamation mark.
3. The Name in the document type declaration must match the element type of the root element.

4.5.2 External DTD

Write a short note on External DTD.

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as **no**. This means, declaration includes information from the external source.

Syntax

Following is the syntax for external DTD :

```
<!DOCTYPE root-element SYSTEM "file-name">
```

where file-name is the file with .dtd extension.

Example

The following example shows external DTD usage :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE book SYSTEM "book.dtd">
<book>
<bname>Food for Life</bname>
<author>Iskon</author>
<price>4500</price>
</book>
```

The content of the DTD file book.dtd are as shown :

```
<!ELEMENT book (bname,author,price)>
<!ELEMENTbname (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

(A) Types of external DTD

There are two types of External DTD :

(a) System Identifier (b) Public Identifier

You can refer to an external DTD by using either system identifiers or public identifiers.

(a) System Identifiers

- A system identifier enables you to specify the location of an external file containing DTD declarations. Syntax is as follows :

```
<!DOCTYPE name SYSTEM "address.dtd" [...]>
```

- It contains keyword SYSTEM and a URI reference pointing to the location of the document.

(b) Public Identifiers

- Public identifiers provide a mechanism to locate DTD resources and are written as follows :

```
<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">
```

- It begins with keyword PUBLIC, followed by a specialized identifier. Public identifiers are used to identify an entry in a catalog. Public identifiers can follow any format, however, a commonly used format is called Formal Public Identifiers, or FPIs.

☞ When to Use a DTD ?

With a DTD, independent groups of people can agree to use a standard DTD for interchanging data.

With a DTD, you can verify that the data you receive from the outside world is valid. You can also use a DTD to verify your own data.

☞ When NOT to Use a DTD ?

XML does not require a DTD. When you are experimenting with XML, or when you are working with small XML files, creating DTDs may be a waste of time. If you develop applications, wait until the specification is stable before you add a document definition. Otherwise, your software might stop working because of validation errors.

Syllabus Topic : XSLT

4.6 XSLT

☞ **What is XSLT? Explain how it works.**

☞ **Discuss the advantages of XSLT.**

XSL (eXtensible Stylesheet Language) is a styling language for XML. XSLT stands for XSL Transformations. XSLT provides the ability to transform XML data from one format to another automatically.

How XSLT Works ?

- An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT style sheet 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.

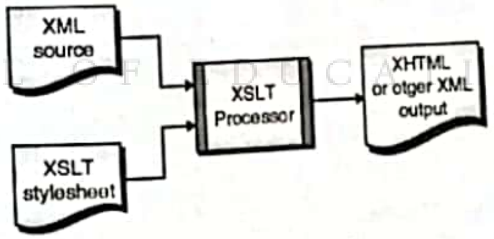


Fig. 4.6.1 : XSLT working

Advantages of XSLT

Here are the advantages of using XSLT :

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

4.6.1 XSLT Syntax

Discuss the steps for creating and executing XSLT.

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

Books.xml

```
<?xml version = "1.0"?>
<class>
<bookbid = "B342">
<bname>Joys Of Life</bname>
<author>Mr. Thadape</author>
<publisher>PVR</publisher>
<price>345</price>
</book>
<bookbid = "B442">
<bname>Food for Life</bname>
<author>Mr. Datar</author>
<publisher>wiley</publisher>
<price>365</price>
</book>
<bookbid = "B974">
<bname>Business Ethics</bname>
<author>Mr. Mane</author>
<publisher>Kitten</publisher>
<price>345</price>
</book>
</class>
```

We need to define an XSLT style sheet document for the above XML document to meet the following criteria –Page should have a title Books Details. Page should have a table of books details. Columns should have following headers Book_ID, BookName, Author, Publisher, Price Table must contain details of the students accordingly.

Step 1 : Create XSLT document

Create an XSLT document to meet the above requirements, name it as books.xsl and save it in the same location where books.xml lies.

Books.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
-<xsl:stylesheetversion="1.0">
-<xsl:template match="/">
-<html>
-<body>
<h2>BOOKS DETAILS</h2>
-<table border="1">
-<trbgcolor="#ffff000">
<th>BOOK ID</th>
<th>BOOK NAME</th>
<th>AUTHOR</th>
<th>PUBLISHER</th>
<th>PRICE</th>
</tr>


-<xsl:for-each select="class/book">
-<tr>
-<td>
<xsl:value-of select="@bid"/>
</td>
-<td>
<xsl:value-of select="bname"/>
</td>


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

```
</td>

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

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

</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

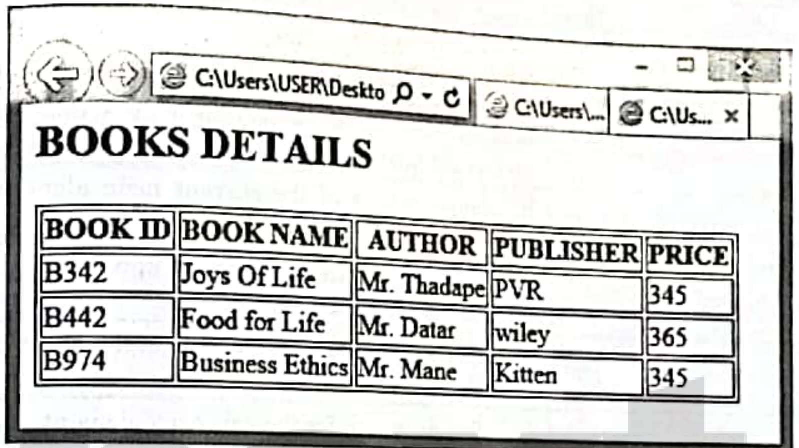
Step 2 : Link the XSLT Document to the XML Document

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

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "Books.xsl"?>
<class>
...
</class>
```

Step 3 : View the XML Document in browser (Internet Explorer).

Output



BOOKS DETAILS

BOOK ID	BOOK NAME	AUTHOR	PUBLISHER	PRICE
B342	Joys Of Life	Mr. Thadape	PVR	345
B442	Food for Life	Mr. Datar	wiley	365
B974	Business Ethics	Mr. Mane	Kitten	345

Syllabus Topic : XSLT Elements and Attributes – xsl:template, xsl:apply-template, xsl:import, xsl:call-template, xsl:include, xsl:element, xsl:attribute-set, xsl:value of

4.7 XSLT Elements Reference

Table 4.7.1 : XSLT Elements

Element	Description
attribute	It adds an attribute
attribute-set	Defines a named set of attributes
apply-imports	Applies a template rule from an imported style sheet
apply-templates	Applies a template rule to the current element or to the current element's child nodes
call-template	Calls a named template
fallback	Specifies an alternate code to run if the processor does not support an XSLT element
for-each	Loops through each node in a specified node set
choose	Used in conjunction with <when> and <otherwise> to express multiple conditional tests

Element	Description
decimal-format	Defines the characters and symbols to be used when converting numbers into strings, with the format-number() function
element	Creates an element node in the output document
copy	It creates a copy of the current node without child nodes and attributes
copy-of	It creates a copy of the current node along with child nodes and attributes
if	Contains a template that will be applied only if a specified condition is true
import	It imports the contents of one style sheet into another.
when	Specifies an action for the <choose> element
with-param	It defines the value of a parameter to be passed into a template
text	It writes text to the output
include	Includes the contents of one style sheet into another.
stylesheet	Defines the root element of a style sheet
template	Rules to apply when a specified node is matched
key	Declares a named key that can be used in the style sheet with the key() function
message	Writes a message to the output.
output	It defines the format of the output document
param	Declares a local or global parameter
sort	Sorts the output
strip-space	Defines the elements for which white space should be removed
namespace-alias	It replaces a namespace in the style sheet to a different namespace in the output
number	It determines the integer position of the current node and formats a number
variable	It declares a local or global variable

Element	Description
transform	It defines the root element of a style sheet
value-of	It extracts the value of a selected node
preserve-space	It defines the elements for which white space should be preserved
processing-instruction	It writes a processing instruction to the output
otherwise	A default action for the <choose> element

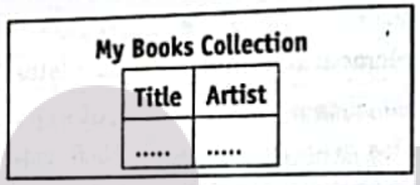
4.7.1 XSLT <xsl:template> Element

- The <xsl:template> element is used to build templates.
- An XSL style sheet consists of several set of rules that are called templates. A template contains rules to apply when a specified node is matched.
- The match attribute of <xsl:template> element is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document. The value of the match attribute is an XPath expression (i.e. match="/" defines the whole document).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0">
<xsl:template match="/">
  <html>
  <body>
  <h2>My Books Collection</h2>
  <table border="1">
   <tr bgcolor="#ffff000">
     <th>Title</th>
     <th>Artist</th>
   </tr>
   <tr>
     <td>....</td>
     <td>....</td>
   </tr>
  </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

- **<xsl:stylesheet>**defines that this document is an XSLT style sheet docum (along with the version number and XSLT namespace attributes).
- The **<xsl:template>** element defines a template. The match="/" attribu associates the template with the root of the XML source document.
- The content inside the **<xsl:template>** element defines some HTML to wri to the output.
- The last two lines define the end of the template and the end of the style shee

Output


My Books Collection

Title	Artist
.....	.....

4.7.2 XSLT <xsl:apply-templates> Element

- The <xsl:apply-templates> element applies a template to the current elem or to the current element's child nodes. If we add a select attribute to th <xsl:apply-templates> element it will process only the child element th matches the value of the attribute. We can use the select attribute to speci the order in which the child nodes are processed.
- Look at the following XSL style sheet :

Books_Data.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type = "text/xsl" href = "Apply_template.xsl"?>
<catalog>
<book>
<title>Me Talk Pretty One Day </title>
<author>Schiward</author>
<price>4589</price>
<year>1965</year>
</book>
<book>
<title>How to Lose Friends and Alienate People </title>
```

```
<author>Wallerd</author>
<price>1244</price>
<year>1982</year>
</book>
<book>
<title>A Thousand Splendid Suns </title>
<author>Williams</author>
<price>678</price>
<year>1990</year>
</book>
</catalog>
```

Apply_template.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0">
<xsl:template match="/">
<html>
<body>
<h2>My Books Collection</h2>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>

<xsl:template match="book">
<p>
<xsl:apply-templates select="title"/>
<xsl:apply-templates select="author"/>
</p>
</xsl:template>

<xsl:template match="title">
 Title: <span style="color:#800000">
<xsl:value-of select="."/></span>
```

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

<xsl:template match="author">
  Artist: <span style="color:#800080">
<xsl:value-of select="."/></span>
<br />
</xsl:template>

</xsl:stylesheet>
```

Run Books_Data.xml with browser.

Output

```
My Books Collection

  Title: Me Talk Pretty One Day
  Artist: Schiward
  Title: How to Lose Friends and Alienate People
  Artist: Wallerd
  Title: A Thousand Splendid Suns
  Artist: Williams
```

4.7.3 XSLT <xsl:import> Element

- The <xsl:import> element is a top-level element that is used to import the contents of one style sheet into another.
- An imported style sheet has lower precedence than the importing style sheet. This element must appear as the first child node of <xsl:stylesheet> or <xsl:transform>.

yntax

```
xsl:import href="URI"/>
```

Attributes

Attribute	Value	Description
href	URI	Specifies the URI of the style sheet to import. It's a required attribute.

Let's see an example: Suppose you have a style sheet called **"Books.xsl"**:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet version = "1.0">
<xsl:template match = "/">
<html>
<body>
<h2>BOOKS DETAILS</h2>
<table border = "1">
<trbgcolor = "#ffff000)">
<th>BOOK ID</th>
<th>BOOK NAME</th>
<th>AUTHOR</th>
<th>PUBLISHER</th>
<th>PRICE</th>
</tr>
<xsl:for-each select="class/book">
<tr>
<td>
<xsl:value-of select = "@bid"/>
</td>
<td><xsl:value-of select = "bname"/></td>
<td><xsl:value-of select = "author"/></td>
<td><xsl:value-of select = "publisher"/></td>
<td><xsl:value-of select = "price"/></td>
</tr>
</xsl:for-each>
   </table>
```

```
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

A second style sheet called "**Books_import.xsl**" imports "Books.xsl":

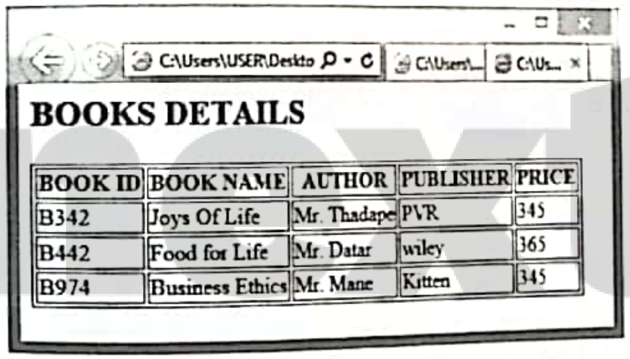
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0">

<xsl:importhref="Books.xsl"/>

<xsl:template match="/">
<xsl:apply-imports/>
</xsl:template>

</xsl:stylesheet>
```

Update **Books_Final.xml** document with the following xml-stylesheet tag. Set href value to **Books_import.xsl**

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "Books.import.xsl"?>
<class>
<book bid = "B342">
<bname>Joys Of Life</bname>
<author>Mr. Thadape</author>
<publisher>PVR</publisher>
<price>345</price>
</book>
<book bid = "B442">
<bname>Food for Life</bname>
<author>Mr. Datar</author>
<publisher>wiley</publisher>
<price>365</price>
```

```
</book>
<book bid = "B974">
<bname>Business Ethics</bname>
<author>Mr. Mane</author>
<publisher>Kitten</publisher>
<price>345</price>
</book>
</class>
```

Output



BOOKS DETAILS

BOOK ID	BOOK NAME	AUTHOR	PUBLISHER	PRICE
B342	Joys Of Life	Mr. Thadape	PVR	345
B442	Food for Life	Mr. Datar	wiley	365
B974	Business Ethics	Mr. Mane	Kitten	345

4.7.4 XSLT <xsl:call-template> Element

It calls a named template.

Syntax

```
<xsl:call-template name="templatename">
  <!-- Content:xsl:with-param* -->
</xsl:call-template>
```

Attributes

Attribute	Value	Description
name	templatename	Specifies the name of the template to be called. It's a required attribute.

Call a template named "detailed_data" when the processor finds a student element :

```
<xsl:template match="student">
  <xsl:call-template name="detailed_data "/>
</xsl:template>
```

4.7.5 XSLT <xsl:include> Element

It is a top-level element that includes the contents of one style sheet into another. An included style sheet has the same precedence as the including style sheet. This element must appear as a child node of <xsl:stylesheet> or <xsl:transform>.

Syntax

```
<xsl:include href="URI"/>
```

Attributes

Attribute	Value	Description
href	URI	Specifies the URI of the style sheet to include. It's a required attribute.

4.7.6 XSLT <xsl:element> Element

It is used to create an element node in the output document.This ability to create both custom elements and attributes, and to display the results, is a major reason why stylesheets generated by XSL are a very sophisticated approach to displaying XML data.

Syntax

```
<xsl:element
name="name"
namespace="URI"
use-attribute-sets="namelist">
  <!-- Content:template -->
</xsl:element>
```

Attributes

Attribute	Value	Description
name	name	Specifies the name of the element to be created (the value of the name attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" />. It's a required attribute.
namespace	URI	Specifies the namespace URI of the element (the value of the namespace attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" namespace="{$someuri}"/>. It's an optional attribute.
use-attribute-sets	namelist	A white space separated list of attribute-sets containing attributes to be added to the element. It's anoptional attribute.

Example

Create a "Writer" element that contains the value of each author element:

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

<xsl:template match="/">
    <xsl:for-each select="catalog/book">
    <xsl:element name="Writer">
    <xsl:value-of select="author" />
    </xsl:element>
    <br />
</xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

4.7.7 XSLT <xsl:attribute> Element

The <xsl:attribute> element is used to add attributes to elements. The <xsl:attribute> element replaces existing attributes with equivalent names.

Syntax

```
<xsl:attribute name="attributename" namespace="uri">
 <!-- Content:template -->
</xsl:attribute>
```

Attributes

Attribute	Value	Description
name	attributename	Specifies the name of the attribute. It's a required attribute.
namespace	URI	Defines the namespace URI for the attribute. It's an optional attribute.

Example

This short example generates an attribute that obtains its value from the XML source. It generates output in the form <IMG src="value-from-XML-source"/>, using an XPath expression that retrieves the appropriate data from the XML source — here, "My_img/images"

```
<IMG>
<xsl:attribute name="src">
<xsl:value-of select="My_img/images" />
</xsl:attribute>
</IMG>
```

Output

```
<IMG src=" My_img/images"/>
```

- While the <xsl:attribute> element can be extremely useful for dynamically creating output attributes that are not known prior to transforming a document, you do not need to use this element if you already know the attributes.

- In the preceding example, you might already know that an IMGelement must contain a src attribute. Because you know this requirement prior to transforming the document, you would not have to use the <xsl:attribute> element.
- You could simplify the transformation syntax and still achieve the same result by using the following :

```
<IMG src="{My_img/images}"/>
```

By using the <xsl:attribute> element instead of attribute value templates, you can :

- Calculate the name of the attribute.
- Use conditionals, templates, and attributes sets in conjunction with attribute generation.
- Add attributes to an element generated by the
  <xsl:copy> or <xsl:element> element.

4.7.8 XSLT <xsl:attribute-set> Element

The <xsl:attribute-set> element creates a named set of attributes. The attribute-set can be applied as whole to the output document. It Must be child of <xsl:stylesheet> or <xsl:transform>.

Syntax

```
<xsl:attribute-set
name="name" use-attribute-sets="name-list">
 <!-- Content:xsl:attribute* -->
</xsl:attribute-set>
```

Attributes

Attribute	Value	Description
name	name	Specifies the name of the attribute-set. It's a required attribute.
use-attribute-sets	name-list	A white space separated list of other attribute-sets to use in the attribute-set. It's an Optional attribute.

Example

Create an attribute-set that can be applied to any output element.

```
<xsl:attribute-set name="font">
  <xsl:attribute name="fname">Times New Roman</xsl:attribute>
  <xsl:attribute name="size">14px</xsl:attribute>
  <xsl:attribute name="color">blue</xsl:attribute>
</xsl:attribute-set>
```

4.7.9 XSLT <xsl:value-of> Element

It is used to extract the value of a selected node. We will refer to our earlie Books.xsl file.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet version = "1.0">
<xsl:template match = "/">
<html>
<body>
<h2>BOOKS DETAILS</h2>

<table border = "1">
<trbgcolor = "#ffff000)">
<th>BOOK ID</th>
<th>BOOK NAME</th>
<th>AUTHOR</th>
<th>PUBLISHER</th>
<th>PRICE</th>
</tr>

<xsl:for-each select="class/book">
<tr>
<td>
<xsl:value-of select = "@bid"/>
```

```
</td>
<td><xsl:value-of select = "bname"/></td>
<td><xsl:value-of select = "author"/></td>
<td><xsl:value-of select = "publisher"/></td>
<td><xsl:value-of select = "price"/></td>
    </tr>
</xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Review Questions

Q. 1 Create a XML file with Internal / External DTD and display it using.

1. CSS
2. XSL

Q. 2 Create a XML file with internal DTD to display the current ticket booking status of particular movie in particular theatre.

Q. 3 Create a XML file with external DTD to display the member details of two cricket teams.

□□□