XML – eXtensible Mark up Language

XML

- It is a new markup language, developed by the W3C (World Wide Web Consortium), mainly to overcome limitations in HTML.
- XML exists because HTML was successful. Therefore, XML incorporates many successful features of HTML. XML also exists because HTML could not live up to new demands. Therefore, XML breaks new ground where it is appropriate.
- XML is unlikely to replace HTML in the near or medium-term. XML does not threaten the Web but introduces new possibilities that are combining XML and HTML. XHTML.

Areas were XML will be a Success now and in near future:

- Large Web site maintenance. XML would work behind the scene to simplify the creation of HTML documents.
- Exchange of information between organizations offloading and reloading of databases.
- Syndicated content, where content is being made available to different Web sites
- Electronic commerce applications where different organizations collaborate to serve a customer.
- Scientific applications with new markup languages for mathematical and chemical formulas.
- Electronic books with new markup languages to express rights and ownership.
- Handheld devices and smart phones with new markup languages optimized for these “alternative” devices.

How is XML different from HTML?

- It has no predefined tags.
- It is stricter
- It is simpler and self describing
- Procedural Markup and generic coding

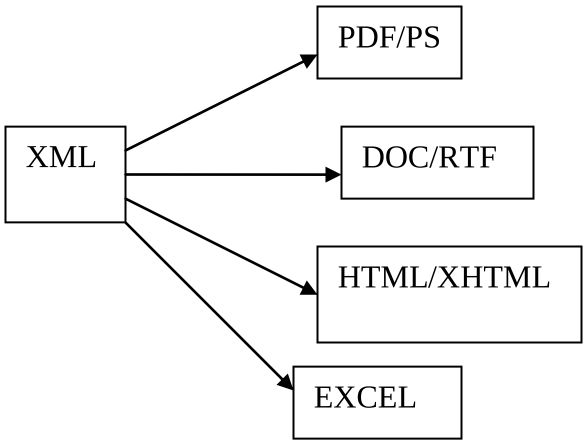
Applications of XML

Applications of XML are classified as being of one of the following two types:

- Document applications manipulate information that is primarily intended for human consumption.
- Data applications manipulate information that is primarily intended for software consumption.

Document Applications of XML

- Converting XML into a publishing format(Display Format)



Data Manipulation Applications:

- Converting XML into data stored in database.

XML Editor:

- XML is textual in nature, so any text editor can be used to write XML program.
- The document consists of *character data* and *markup*. Both are represented by text. Ultimately, it's the character data we are interested in because that's the information. However, the markup is important because it records the structure of the document. There are a variety of markup constructs in XML but it is easy to recognize the markup because it is always enclosed in angle brackets.

Conversion of XML into different formats:

- XML is converted in document/data formats(using XML Parser) by following ways:
  - ❖ XSL Processor.
  - ❖ DOM Parser.
  - ❖ SAX Parser.

Getting started with XML

- The building block of XML is Element. Each Element has a name and content is given with angle brackets.

Rules for naming Elements:

- Name should start with an alphabet (uppercase/lowercase) or an underscore
- Special characters like – (hyphen), .(dot) are included besides underscore.
- Name can have number in between and at the end, but not at the start.
- Some Valid element names-**<story>,<story.tell>,<story1>,<story.1>,<story-1>**

Rules of XML (Syntax):

- XML elements must be **properly nested**
- XML elements and attributes should follow the rules of naming elements.
- XML elements must always be **closed** (start tag should have a end tag)

- XML documents must have **one root element.**
- XML attribute values must be quoted.

Store the XML file with .xml extension

Sample XML Codes:

Example 1: XML with Element and content

```
<? xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="newstylesheet.xsl"?>

<Personnel>

 <Employee>
      <Name>S.Hemalatha</Name>
      <Id>3674</Id>
      <Age>26</Age>
  </Employee>

 </Personnel>
```

Elements in the above example are Personnel (root), Employee, Name, Id, and Age. The text enclosed within the start and end tag is content of the element. The content here is S.Hemalatha, 3674, 26.

Example 2: XML with element, content and attribute (type)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="newstylesheet.xsl"?>
<Personnel>
 <Employee type="Permanent">
      <Name>S.Hemalatha</Name>
      <Id>3674</Id>
      <Age>26</Age>
  </Employee>
 <Employee type="Permanent">
      <Name>C.Ranichandra</Name>
      <Id>3675</Id>
      <Age>25</Age>
   </Employee>
 <Employee type="Temporary">
      <Name>N.C.Senthil Kumar</Name>
      <Id>3676</Id>
      <Age>28</Age>
   </Employee>

</Personnel>
```

When the XML file is opened in DTD (discussed in next section) aware editor, the display will be different format. If opened in non DTD aware editor, it will be displayed in plain text.

Well Formed Document:

- XML document is well formed, if it follows all the rules (Syntax) of the Language.

Valid Document

- XML document is valid, if follows all the rules (Syntax) of the Language and also conforms the corresponding DTD.
- A valid document is well formed, but well formed document need not be valid.

Explanation of the lines of code of a XML Program:

XML Declaration:

- The *XML declaration* is the first line of the document. The declaration identifies the document as an XML document. The declaration also lists the version of XML used in the document. Apart from that it gives information about the type of encoding used.

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

- **standalone attribute:**
  It indicates whether a document requires a Document Type Definition (DTD) in order to be rendered correctly.
  If XML document requires a DTD, then give the standalone attribute a value “no” else give a value “yes”.
- **Is DTD Mandatory? If so Why/Why Not?**
  No DTD is not mandatory as XML is self describing, it is useful in PDAs and mobile phones that don’t have a power to process DTDs.It can be used in XML documents, if multiple XML documents shares same elements.

Style Sheet Inclusion:

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

XML can be published in any format, for which it requires XSL, DOM Parsers, and SAX Parsers. This PI (Processed Instruction) gives information that The XML document is translated into publishing format using XSL **newstylesheet.xsl.**

Following these two PIs are user defined tags used to define Employee information.

The **root** element is **Personnel**.**Employee**, **Name, and Age and**, **Id** are **elements** of XML document and **type** is an attribute.

The XML document given above is **Well Formed Document, but** need not be a Valid document as it is not checked against the DTD.

Is Usage of attributes Mandatory?

No.Attributes can be converted into Elements.

```
<Employee type="Permanent">
     <Name>S.Hemalatha</Name>
     <Id>3674</Id>
     <Age>26</Age>
  </Employee>
```

Can be converted into
```
<Employee >
      <Name>S.Hemalatha</Name>
     <type>Permanent</type>
      <Id>3674</Id>
      <Age>26</Age>
  </Employee>
```
Converting attributes into Elements helps in increasing the Node tree as type can be a parent to any other element.This can't be done with attributes.

- **Element Employee is the parent of Name, Id, and Age and, now type.Element Personnel is the Parent of Employee.**

Try Answering Simple Questions?

1. XML is based on ______________ Language.
2. In XML, the information contained within < > pair is________________.
3. In XML, the information containing within start tag and end tag is _________.
4. XML defines elements and attributes on it's own(true/false)-
5. XML declaration that doesn't require a DTD in order to be rendered correctly is given as______________________________________.
6. Comments in XML are given as________________________.
7. When a XML document adheres to XML's Syntax rules, the document is called__________.
8. A program that only checks if a XML document is well formed is ___________.
9. A program that not only checks if a XML document is well formed but also valid is ___________.
10. ______________element contains all the other elements of a page.

Example 3: Film XML
```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="flims.xsl"?>
<flims-dairy>

    <flim category="commercial"  language="Tamil">
       <name>Dalapathi</name>
       <actor_1>Rajni</actor_1>
       <actor_2>Mamooty</actor_2>
       <director> Mani Rathnam</director>
     </flim>

   <flim category="art" language="Hindi">
     <name> A Wednesday</name>
     <actor_1>Naser</actor_1>
     <actor_2>Anupam Kher</actor_2>
     <director> Neeraj Pandey</director>
   </flim>

  <flim category="art"  language="Tamil">
      <name>Moondaram Pirai</name>
      <actor_1>Kamal Hasan</actor_1>
```

```
        <actor_2>Sridevi</actor_2>
        <director>Balu Mahendra</director>
   </flim>

  <flim category="art"  language="Hindi">
        <name>Mirch Masala</name>
        <actor_1>Naser</actor_1>
        <actor_2>Smita Patel</actor_2>
        <director>Ketan Mehta</director>
  </flim>

   <flim category="commercial"  language="English">
        <name>Titanic</name>
        <actor_1>Leonardo DiCaprio</actor_1>
        <actor_2>Kate Winslet</actor_2>
        <director>James Cameron</director>
   </flim>


   <flim category="classic"  language="Hindi">
        <name>Chori Chori</name>
        <actor_1>Nargis</actor_1>
        <actor_2>Raj Kapoor</actor_2>
        <director>Anant Thakur</director>
   </flim>
   <flim category="classic"  language="English">
        <name>Gone With The Wind</name>
        <actor_1>Vivien Leigh</actor_1>
        <actor_2>Clarke Gable</actor_2>
        <director>Victor Fleming</director>
   </flim>
  <flim category="commercial"  language="English">
        <name>Step Mom</name>
        <actor_1>Julia Roberts</actor_1>
        <actor_2>Susan Saradon</actor_2>
        <director>Chris Coloumbus</director>
   </flim>
   <flim category="commercial"  language="Hindi">
        <name>DDLJ</name>
        <actor_1>Sharukh Khan</actor_1>
        <actor_2>Kajol</actor_2>
        <director>Aditya Chopra</director>
   </flim>
</flims-dairy>
```

Example 4: BOOK XML using Empty elements.

```
<?xml version="1.0"?>
<!DOCTYPE books SYSTEM "book.dtd">
<?xml-stylesheet type="text/xsl" href="books.xsl"?>
<books>
  <book name="Coma" author="Robin Cook" />
  <book name="The Prodigal Daughter" author="Jeffery Archer" />
  <book name="Not A Penny More, Not A Penny Less" author="Jeffery Archer"
/>
  <book name="Love Story" author="Erich Segal" />
  <book name="Master Of The Game" author="Sidney Sheldon" />
  <book name="Pride and Prejudice" author="Jane Auston" />
  <book name="Gone With The Wind" author="Margaret Mitchell" />

</books>
```

book.dtd

```
<!ELEMENT books (book)*>

<!ELEMENT book EMPTY>
<!ATTLIST book
   author CDATA #IMPLIED
   name CDATA #IMPLIED
  >
```

DOCUMENT TYPE DEFINITION (DTD)

DTD:

- It is a set of rules that define the elements and attributes that are defined in XML document.
- It also defines how the elements should be structured in XML documents.
- An XML document should contain only those elements and attributes that are defined in DTD and be structured according to the DTD's rules, or it is not valid...

Valid Document

- XML document is valid, if follows all the rules (Syntax) of the Language and also conforms the corresponding DTD.
- A valid document is well formed, but well formed document need not be valid.

Non Validating and Validating Parsers.

- **Non Validating Parsers:** Checks if the XML document is well formed.Eg-browsers
- **Validating Parsers:** Checks if XML document is valid.

DTD
<!DOCTYPE> tag is used to create DTD's.There are two types of DTDs:Internal and External.An Internal DTD is defined within an XML Doument and an External DTD is defined in a separate document with and extension of .dtd.

How is External DTD different from Internal DTD?

- XML declaration and document type declaration is not included in external DTD. The definitions are not included within [].

```
<?xml version="1.0"?>
<!DOCTYPE entry SYSTEM "address-book.dtd">
<entry>
<name>Amirtha Sathish</name>
<address>
<street>111,PCK Nagar</street>
<city>Katpadi</city>
<pin-code>45202</pin-code>
<state>Tamil Nadu</state>
<country>India</country>
</address>
<tel preferred ="true" >513-555-8889</tel>
<tel>513-555-7098</tel>
<email href = "mailto:ammu@gmail.com"/>
</entry>
```

DTD file- **address-book.dtd**

```
<!ELEMENT entry (name,address,telephone,email)*>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (street,city,pin-code,state,country)*>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT pin-code (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ATTLIST tel
   preferred CDATA #IMPLIED
  >
<!ELEMENT email EMPTY>
<!ATTLIST email
   href CDATA #IMPLIED
  >
```

The external DTD is not stored in the document. It is referenced from the document type declaration through an identifier in the previous example:

<!DOCTYPE entry SYSTEM "address-book.dtd">

Document Type Declaration

The *document type declaration* attaches a DTD to a document. Don't confuse the document type declaration with the document type definition(DTD).

The document type declaration has the form:
<!DOCTYPE address-book SYSTEM "address-book.dtd">

Identifiers

- There are two types of identifiers: system identifiers and public identifiers.A keyword, respectively **SYSTEM** and **PUBLIC**, indicates the type of identifier.
- A system identifier is a Universal Resource Identifier (URI) pointing to the DTD. URI is a superset of URLs
- A public identifier points to a DTD recorded with the ISO according to the rules of ISO 9070.
- The system identifier is easy to understand. The XML processor must download the document from the URI.
- Public identifiers are used to manage local copies of DTDs. The XML processor maintains a catalog file that lists public identifiers and their associated URIs. The processor will use these URIs instead of the system identifier.

Advantages of having local copies:

- If the URIs in the catalog point to local copies of the DTD, the XML processor saves some downloads.

Internal DTD for the same- defined within XML Document itself

```
<? xml version="1.0"?>
<!DOCTYPE entry [
<!ELEMENT entry (name,address,telephone,email)*>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (street,city,pin-code,state,country)*>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT pin-code (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ATTLIST tel
   preferred CDATA #IMPLIED
  >
<!ELEMENT email EMPTY>
<!ATTLIST email
   href CDATA #IMPLIED
  >]
<entry>
<name>Amirtha Sathish</name>
```

```
<address>
<street>111,PCK Nagar</street>
<city>Katpadi</city>
<pin-code>45202</pin-code>
<state>Tamil Nadu</state>
<country>India</country>
</address>
<tel preferred ="true" >513-555-8889</tel>
<tel>513-555-7098</tel>
<email href="mailto:ammu@gmail.com"/>
</entry>
```

An XML Document can have both internal and external DTDs.

Public Identifiers Format

The following public identifiers point to the address book:
<!DOCTYPE address-book PUBLIC **“-//VIT//Address Book//EN”>**
There are four parts, separated by “//”:

- The first character is + if the organization is registered with ISO,-otherwise
- The second part is the owner of the DTD.
- The third part is the description of the DTD; spaces are allowed.
- The final part is the language (EN for English).

Benefits of the DTD

The main benefits of using a DTD are

- The XML processor enforces the structure, as defined in the DTD.
- The application accesses the document structure, such as to populate
- an element list.
- The DTD gives hints to the XML processor—that is, it helps separate indenting from content.
- The DTD can declare default or fixed.

Element Declaration

DTD is a mechanism to describe every object (element, attribute, and so on) that can appear in the document, starting with elements.
The following is an example of element declaration:
<!ELEMENT flims-dairy (flim+)>

- After the <!ELEMENT markup comes the element name followed by its content model. The element declaration is terminated with a right angle bracket. Element declarations are easy to read: The right side (the content model) defines the left side (the element name). In other words, the content model lists the children that are acceptable in the element.
- Parentheses() are used to group elements in the content model

Special Keywords

For most elements, the content model is a list of elements. It also can be one of the following keywords:

- #PCDATA stands for parsed character data and means the element can contain text. #PCDATA is often (but not always) used for leaf elements. Leaf elements are elements that have no child elements.
- EMPTY means the element is an empty element. EMPTY always indicates a leaf element.
- ANY means the element can contain any other element declared in theDTD. This is seldom used because it carries almost no structure information.ANY is sometimes used during the development of a DTD,before a precise rule has been written.
- Element contents that have #PCDATA are said to be mixed content. Element contents that contain only elements are said to be element content.
  <!ELEMENT tel (#PCDATA)>
- <!ELEMENT email EMPTY>

Plus, Star, and Question Mark in DTD

The plus (“+”), star (“*”), and question mark (“?”) characters in the element content are occurrence indicators. They indicate whether and how elements in the child list can repeat.

- An element followed by no occurrence indicator must appear once and only once in the element being defined.
- An element followed by a “+” character must appear one or several times in the element being defined. The element can repeat.
- An element followed by a “*” character can appear zero or more times in the element being defined. The element is optional but, if it is included, it can repeat indefinitely.
- An element followed by a “?” character can appear once or not at all in the element being defined. It indicates the element is optional and, if included, cannot repeat.

The entry and name elements have content model that uses an occurrence indicator:

<!ELEMENT entry (name,address*,tel*,fax*,email*)>
<!ELEMENT address (street,city?,pin-code,state,country)>

Acceptable children for the entry are name, address, tel, fax, and email.Except for name, these children are optional and can repeat.Acceptable children for address are street, region, pin-code, locality,and country. None of the children can repeat but the region is optional.

Comma and Vertical Bar

The comma (“,”) and vertical bar (“|”) characters are connectors.Connectors separate the children in the content model, they indicate the order in which the children can appear. The connectors are

- The “,” character, which means both elements on the right and the left of the comma must appear in the same order in the document.
- The “|” character, which means that only one of the elements on the left or the right of the vertical bar must appear in the document.

Attributes:

Attributes also must be declared in the DTD. Element attributes are declared with the ATTLIST declaration, for example:
<!ATTLIST tel preferred (true | false) "false">

The various components in this declaration are the markup (<!ATTLIST), the element name (tel), the attribute name (preferred), the attribute type ((true | false)), a default value (“false”), and the right angle bracket.For elements that have more than one attribute, you can group the declarations.For example, email has two attributes:
<!ATTLIST email href CDATA #REQUIRED preferred (true | false) "false">
Attribute declaration can appear anywhere in the DTD. For readability, it is best to list attributes immediately after the element declaration.
The DTD can specify a default value for the attribute. If the document does not include the attribute, it is assumed to have the default value. The default value can take one of the four following values:

- #REQUIRED means that a value must be provided in the document.

  <!ELEMENT price (#PCDATA)>
  <!ATTLIST price currency NMTOKEN #FIXED "Rupees">
  NMTOKEN is word without spaces.
  <price>19.99</price> in a document, appears as though it reads
  <price currency="Rupees">19.99</price>

- #IMPLIED means that if no value is provided, the application must use its own default
- #FIXED followed by a value means that attribute value must be the value declared in the DTD
- A literal value means that the attribute will take this value if no value is given in the document.

XSL Transformation:

- A XML Stylesheet language.
- Language used to convert XML documents in XML (with a different DTD), HTML, and other formats
- Publish a large set of documents.
- Extract information from XML documents.

Why Styling?

XML concentrates on the structure of the information and not its appearance.However, to view XML documents we need to format or style them.Obviously, the styling instructions are directly related to and derived from the structure of the document.In practice, styling instructions are organized in style sheets; to view a document,apply the appropriate style sheet to it.The W3C has published two recommendations for style sheets: CSS, short for Cascading Style Sheet, and XSL, short for XML Stylesheet Language.

CSS

CSS was originally developed for HTML and browsers that support XML also largely support it. A CSS is a set of rules that tells the browser which font, style, and margin to use to display the text, as shown in the following example:

```
p.ind
{
font-family: Times New Roman;
font-size: 10pt;
margin: 5px;
display: block;
font-style: italic
}
```

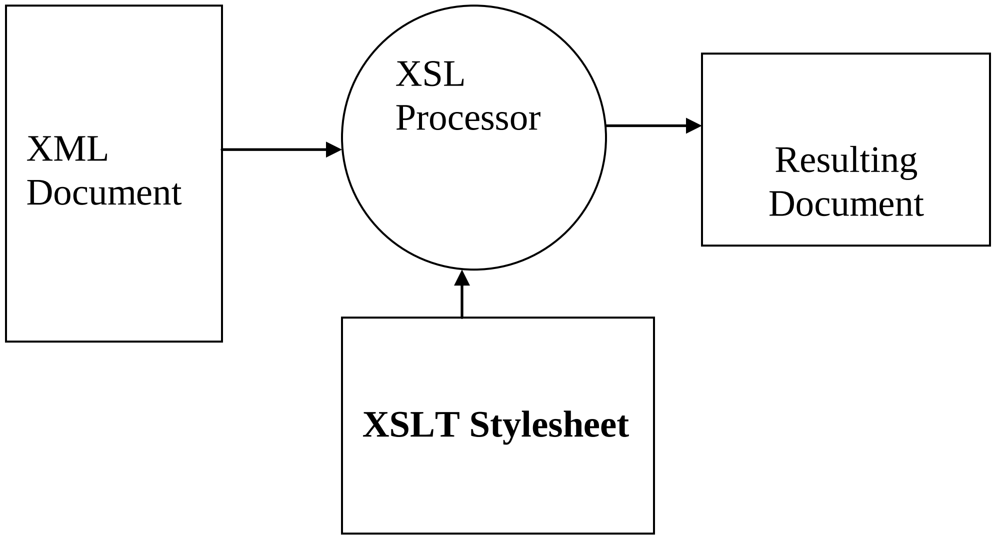
The styling is applied to a para p with a class ind.

XSL(T)

XSL is more ambitious than CSS because it supports transforming the document before display. XSL would typically be used for advanced styling.

How XSL(T) Works?

XSLT is a language to specify transformation of XML documents. It takes an XML document and transforms it into another XML document/HTML/other formts. This is illustrated in the figure below.



What Does a XSL give to is users?

- Add elements specifically for viewing, such as add the logo or the address of the sender to an XML invoice create new content from an existing one, such as create the table of contents
- Present information with the right level of details for the reader,such as using a style sheet to present high-level information to amanagerial person while using another style sheet to present more detailed technical information to the rest of the staff.
- Convert between different DTDs or different versions of a DTD, such as convert a company specific DTD to an industry standard.
- Transform XML documents into HTML for backward compatibility with existing browsers.
- XSL enables programmers to write the document in one format (XML) and automatically create distribution copies in text and HTML.
- Furthermore, because styling is applied automatically, it is easy to change the layout of the Web site by just changing the style sheet. As Web fashion keeps changing, this is a major advantage.

Viewing XML Document

Try opening your XML in WORD,BROWSER,WORDPAD and EXCEL.Enjoy what you view.

Style Sheet Programs: Converting XML into HTML.Save XSL file with .xsl extension.

Style sheet for Example 2 XML Document.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
    <xsl:output method="html"/>
   <xsl:template match="Personnel">
        <html>
            <head>
                <title>Employee Dairy</title>
            </head>
            <body>
                    <xsl:apply-templates/>
               </body>
        </html>
    </xsl:template>

<xsl:template match="Employee">
<p><xsl:apply-templates/></p>
</xsl:template>

<xsl:template match="Name">
<i><xsl:apply-templates/></i>
</xsl:template>

<xsl:template match="Id">
<i><xsl:apply-templates/></i>
</xsl:template>

<xsl:template match="Age">
<i><xsl:apply-templates/></i>
</xsl:template>

</xsl:stylesheet>
```

<u>XSL file for Example 3 XML Document</u>

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method="html"/>
<xsl:template match="/">
        <html>
            <head>
                <title>flims.xsl</title>
            </head>
            <body>
               <center> <h1>Film Dairy</h1></center>
                <table border="1" align="center">
                <thead>
                  <tr>
                       <td>Name</td>
                       <td>Category</td>
                       <td>Language</td>
                       <td>Actor</td>
                       <td>Actor</td>
                       <td>Director</td>
                  </tr>
                </thead>
                <tbody>
                       <xsl:for-each select="flims-dairy/flim">
                             <tr>
                                 <td><xsl:value-of select="name"/></td>
                                 <td><xsl:value-of select="@category"/></td>
                                 <td><xsl:value-of select="@language"/></td>
                                 <td><xsl:value-of select="actor_1"/></td>
                                 <td><xsl:value-of select="actor_2"/></td>
                                 <td><xsl:value-of select="director"/></td>
                             </tr>
                        </xsl:for-each>
                </tbody>
             </table>
            </body>
        </html>
    </xsl:template>

</xsl:stylesheet>
```

<u>XSL file for Example 4 XML Document</u>

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method="html" encoding="UTF-8"/>
<xsl:template match="/">
<html>
<head><title>Books</title>
</head>
<body>
<table width="100%" border="1" bgcolor="orange">
  <thead>
          <tr>
            <td width="50%"><h2>Name Of The BOOK</h2></td>
            <td width="50%"><h2>Author</h2></td>
          </tr>
  </thead>
  <tbody>
          <xsl:for-each select="books/book">
          <tr>
            <td width="50%"><h3><xsl:value-of select="@name" /></h3></td>
            <td width="50%"><h3><xsl:value-of select="@author" /></h3></td>
          </tr>
          </xsl:for-each>
  </tbody>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Style Sheet Elements:

The top level element is xsl:stylesheet

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

XSLT Elements

The bulk of the style sheet is a list of templates.

The <xsl:template> Element

The <xsl: template> element is used to build templates.

The **match** attribute 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, matching every element of XML Document for which the XSL is written).

The following XSL code,transforms the title of a section in an HTML paragraph with the text in
italic:

```
<xsl:template match="Employee">
<p><xsl:apply-templates/></p>
</xsl:template >
```

A template has two parts:

The match parameter is a path to the element in the source tree to which the template applies.

The content of the template lists the elements to insert in the resulting tree.(content is any text(plain/html) between start tag and end tag).

<p><xsl:apply-templates/></p> is the content of xsl:template.

Paths

The syntax for XML paths is similar to file paths. XML paths start from the root of the document and list elements along the way. Elements are separated by the "/" character.
The root of the document is "/". The root is a node that sits before the toplevel element. It represents the document as a whole.
flims-dairy/flim denotes flim element which is present within flims-dairy.

The <xsl:apply-templates>

The <xsl:apply-templates> element applies a template to the current element or to the current element's child nodes.

The <xsl:value-of> Element

The <xsl:value-of> element can be used to extract the value of an XML element and add it to the output stream of the transformation:

The <xsl:for-each> Element

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

- Create a XML for QUIZ.Create a XSL that transforms this XML into html,where each quiz question becomes a radio button control.
- Include a Javascipt in the html file and process the quiz and gives out the result.