4.5 .XML DOM and XML PARSERS

The Document Object Model (DOM) is the foundation of XML. XML documents have a hierarchy of informational units called nodes; DOM is a way of describing those nodes and the relationships between them.

DOM

A DOM Document is a collection of nodes or pieces of information organized in a hierarchy. This hierarchy allows a developer to navigate through the tree looking for specific information.

```
<!DOCTYPE html> <html><body>
<h1> DOM example </h1>
<div> <b>Name:</b><span id="name"></span><br>
<b>Company:</b><span id="company"></span><br>
<b>Phone:</b><span id="phone"></span> </div>
<script> if (window.XMLHttpRequest)
{// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp = new XMLHttpRequest(); }
```

```
else {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP"); }
xmlhttp.open("GET","/xml/address.xml",false);
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;
document.getElementById("name").innerHTML=
xmlDoc.getElementsByTagName("name")[0].childNodes[0].nodeValue;
document.getElementById("company").innerHTML=
xmlDoc.getElementsByTagName("company")[0].childNodes[0].nodeValue;
document.getElementById("phone").innerHTML=
xmlDoc.getElementsByTagName("phone")[0].childNodes[0].nodeValue;
</script></body></html>
```

address.xml

```
<?xml version="1.0"?>
<contact-info>
<name>Sharanya</name>
<company>abc</company>
<phone>(011) 123-4567</phone>
</contact-info>
```

XML PARSERS

XML parser is a software library or a package that provides interface for client applications to work with XML documents. It checks for proper format of the XML document and may also validate the XMLdocuments.

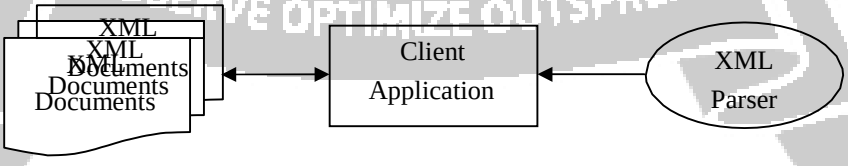

Fig 4.2 XML Parsers

The goal of a parser is to transform XML into a readable code. To ease the process of parsing, some commercial products are available that facilitate the breakdown of XML document and yield more reliable results.

VALIDATION

An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration (DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are: Well-formed XML document and Valid XML document

- **Well-formed XML document**
  - Non DTD XML files must use the predefined character entities for amp(&), apos (single quote), gt >), lt (<), quot (double quote).
  - It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
  - Each of its opening tags must have a closing tag or it must be a self- ending tag.(<title> ...</title> or <title/>).
  - It must have only one attribute in a start tag, which needs to be quoted.
  - Amp (&), apos (single quote), gt (>), lt (<), quot (double quote) entities other than these must be declared.

```
<?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>Sharanya</name>
<company>abc</company>
<phone>(011) 123-4567</phone> </address>
```

- **Valid XML document**

  If an XML document is well-formed and has an associated Document Type Declaration (DTD), then it is said to be a valid XML document.

XSL (XML Style Sheet)

XML concentrates on the structure of the information and not its appearance. The W3C has published two recommendations for style sheets: CSS (Cascading Style Sheet) and XSL(XML Style sheet Language).XSL supports transforming the document before display. XSL would typically be used for advanced styling. XSL originally consisted of three parts:

- XSLT (XSL Transformation) - a language for transforming XML documents
- XPath - a language for navigating in XML documents
- XSL-FO (XSL Formatting Objects) - a language for formatting XML documents

XSL

```
<P><B>Table of Contents</B></P> <UL>
<xsl:for-each select="article/section/title">
<LI><A><xsl:value-of select="."/></A></LI>
</xsl:for-each> </UL>
```

XSLT (XSL Transformation)

XSLT is a language to specify transformation of XML documents. It takes an XML document and transforms it into another XML document. XSLT is an XML-related technology that is used to manipulate and transform XML documents.

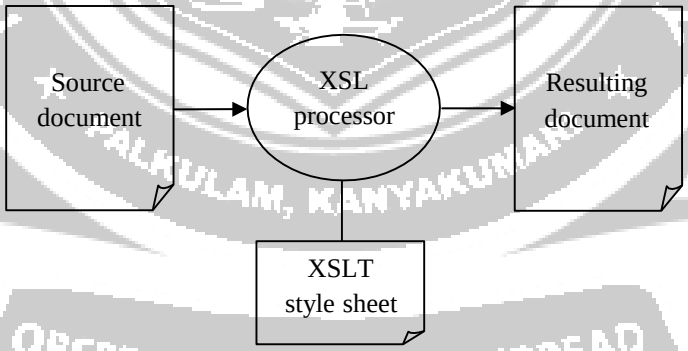

Fig 4.2 XSLT Transformation

With XSLT, the user can take an XML document and choose the elements and values, then generate a new file with new choices. Because of XSLT's ability to change the content of an XML document, XSLT is referred to as the **stylesheet for XML**. XSLT is not limited to styling activities. Many applications require transforming documents. XSLT can be used to:

- 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 a managerial 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.

XSLT

XML code	XSLT code
<?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet type="text/xsl" href="class.xsl"?> <class> <student>Arthi</student> <student>Ambarish</student> <student>Anitha</student> <teacher>Sharanya</teacher> </class>	<?xml version="1.0" ?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" > <xsl:template match="teacher"> <u><xsl:value-of select="."/></u> </xsl:template> <xsl:template match="student"> <xsl:value-of select="."/> </xsl:template> <xsl:template match="/"> <html><body> <xsl:apply-templates/> </body></html> </xsl:template></xsl:stylesheet>

The XML file class.xml is linked to the XSLT code by adding the xml-stylesheet reference. The XSLT code then applies its rules to transform the XML document.

- Before XSLT: classoriginal.xml
- After XSLT rules are applied: class.xml

XSLT Syntax

➢ **XSLT - XML Declaration**

The user includes an XML declaration at the top of the XSLT documents. The attribute version defines what version of XML is used.

Example: <?xml version="1.0" ?>

- **XSLT - Stylesheet Root Element**

  Every XSLT file must have the root element xsl:stylesheet. This root element has two attributes that must be included:

  - version - the version of XSLT
  - xmlns:xsl - the XSLT namespace, which is a URI to w3.org

- **XSLT - XSL: Namespace Prefix**

  The root element specifies the XSL namespace. The standard form of an XSL element is: xsl:element

XSLT - Stylesheet Reference

Linking XML document to XSLT stylesheet is stylesheet reference. This is the magic step that connects XML to a XSLT file

XSLT - xml-stylesheet

xml-stylesheet is a special declaration in XML for linking XML with stylesheets. Place this after XML declaration to link the XML file to the XSLT code. xml-stylesheet has two attributes:

- type: the type of file being linked to. We will be using the value text/xsl to specify XSLT.
- href: the location of the file. If the user saved the user XSLT and XML file in the same directory, the user can simply use the XSLT filename.

Make sure that both XSLT and XML file are in the same directory.

Reference

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="class.xsl"?>
<class>          <student>Arun</student>
        <student>Divya</student>
        <teacher>Sharanya</teacher>  </class>
```

XSLT: XSL Template

The purpose of XSLT is to help transform an XML document into something new. To transform an XML document, XSLT must be able to do two things well:

- Find information in the XML document.
- Add additional text and/or data.

Both of these items are taken care of with the very important XSL element xsl:template.

XSLT - xsl:template Match Attribute

To find information in an XML document use xsl:template's match attribute. It is in this attribute the knowledge of XPath is used to find information in the XML document. In previous example, to find student elements, we would set the match attribute to a simple XPath expression: student.

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="student">
                Found a learner!
        </xsl:template>
<stylesheet>
```

XSLT - xsl:apply-templates

The xsl:apply-templates element to be more selective of the XML data.

- **XSLT - Remove Unwanted Text**

The following attributes are used to remove unwanted text:

- select attribute: lets the user choose specific child elements
- xsl:apply-templates: to decide when and where the xsl:template elements are used

XSLT - Remove Unwanted Children

We could use the select attribute to select specific child elements. To do this, we need a new xsl:template that matches our XML document's root element, class. We can then pick the child student using the select attribute. Here's the XSLT code to get the job done.

apply templates

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="class">
<xsl:apply-templates select="student"/>
</xsl:template>
<xsl:template match="student">
```

```
Found a learner!
</xsl:template></xsl:stylesheet>
```

Found a learner! Found a learner! Found a learner!

The XSLT processor begins at the root element when looking for template matches. Because we have a match for the root element, class, the code we just added is used first.

xsl:apply-templates

In our template that matched class, we use xsl:apply-templates which will check for template matches on all the children of class. The children of class in our XML document are student and teacher.

xsl:apply-templates select="student"

To have the teacher element, "Sharanya," ignored, we use the select attribute of xsl:apply-templates to specify only student children.

The XSLT processor then goes searching templates that only match student elements.

xsl:template match="student"

The processor finds the only other template in our XSLT, which prints out, "Found a learner!" for each student element in the XML document. XSLT finds three students, so "Found a learner!" is displayed three times.

XSLT - Well-Formed Output

To obtain well formed output remove the root element in an XSLT template and inserting a new root element for the output. To do this, we are going to need to add an <html> (root element) tag, a <body> tag, and maybe some <p> tags.

XSLT - Replacing the Old Root Element

In the template that matches the original root element, we will insert the <html> tag to be the output's root element. We can also put the <body> tag there. In the template that matches the student elements, we can insert a <p> tag to make a separate paragraph for each student.

Replacing root

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="class"> <html><body>
<xsl:apply-templates select="student"/> </body></html> </xsl:template>
```

```
<xsl:template match="student">
<p>    Found a  learner!</p>  </xsl:template></xsl:stylesheet>
```

```
<html><body>
<p>Found a learner!</p>
<p>Found a learner!</p>
<p>Found a learner!</p>
</body></html>
```

4.6 NEWSFEEDS and ATOM

NEWS FEED

News feeds are an example of automated syndication. News feed technologies allow information to be automatically provided and updated on Web sites, emailed to users, etc. As the name implies news feeds are normally used to provide news; however the technology can be used to syndicate a wide range of information**.**

The **BBC ticker** is an example of a news feed application. A major limitation with this approach is that the ticker can only be used with information provided by the BBC. The RSS (Really Simple Syndication)standard was developed as an open standard for news syndication, allowing applications to display news supplied by any RSS provider.

RSS (Really Simple Syndication)

RSS is an XML dialect used to publish frequently updated content, such as blog posts or news headlines.

It is a way to easily distribute a list of headlines, update notices, and sometimes content to a wide number of people. It is used by computer programs that organize those headlines and notices for easy reading.

The content feed is identified by a unique URI, and this URI is used by an RSS Reader application to retrieve and display the content feed from a web site. An RSS feed can contain one or more images or items. An item can be a synopsis of an article with a link to the full article or the entire article itself. An RSS has a well-defined structure. Some Web APIs use RSS as the data format returned when a request is made.

Working of RSS

RSS works by having the website author maintain a list of notifications on their website in a standard way. This list of notifications is called an **RSS Feed**. People who are interested in finding out the latest headlines or changes can check this list. Special computer programs called **RSS aggregators** have been developed that automatically access the RSS feeds of websites of user's interest and organize the results.

Producing an RSS feed is very simple and hundreds of thousands of websites now provide this feature, including major news organizations like the New York Times, the BBC, and Reuters, as well as many weblogs.

RSS provides very basic information to do its notification. It is made up of a list of items presented in order from newest to oldest. Each item usually consists of a simple title describing the item along with a more complete description and a link to a web page with the actual information being described. Sometimes this description is the full information the user want to read (such as the content of a weblog post) and sometimes it is just a summary.

Creating RSS feed

The special XML-format file that makes up an RSS feed is usually created in one of a variety of ways. Most large news websites and most weblogs are maintained using special **content management** programs. Authors add their stories and postings to the website by interacting with those programs and then use the program's publish facility to create the HTML files that make up the website. Those programs often also can update the RSS feed XML file at the same time, adding an item referring to the new story or post, and removing less recent items.

Blog creation tools like Blogger, LiveJournal, Movable Type, and Radio automatically create feeds. Websites that are produced in a more custom manner, such as with Macromedia Dreamweaver or a simple text editor, usually do not automatically create RSS feeds. Authors of such websites either maintain the XML files by hand, just as they do the website itself, or use a tool such as Software Garden, Inc.'s ListGarden program to maintain it.

There are also services that periodically read requested websites themselves and try to automatically determine changes (this is most reliable for websites with a somewhat regular news-like format), or that let the users create RSS feed XML files that are hosted by that service provider.

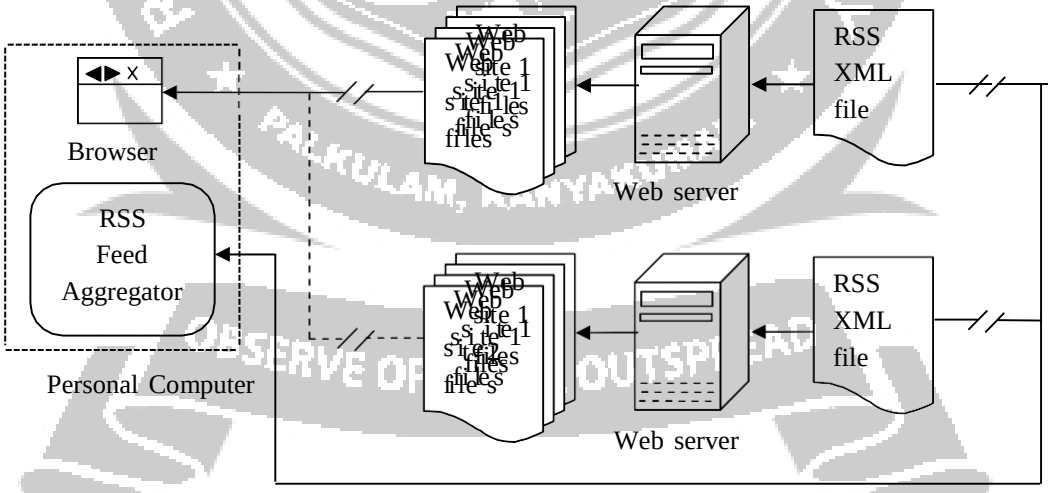

Fig 4.3: Communication between websites, RSS feed and PC

In the above diagram, a web browser being used to read first Web Site 1 over the Internet and then Web Site 2. It also shows the RSS feed XML files for both websites being monitored simultaneously by an RSS Feed Aggregator.

- **Sections of an RSS file**

Apart from the root element there are four main sections of the RSS file. These are the channel, image, item, and text input sections. In practical use, the channel and item elements are requirements for any useful RSS file, while the image and text input are optional.

The channel section

The channel element contains metadata that describe the channel itself, telling what the channel is and who created it. The channel is a required element that includes the name of the channel, its description, its language, and a URL. The URL is normally used to point to the channel's source of information.

Channel element

```
<channel><title>MozillaZine</title>
<link>http://www.mozillazine.org</link>
<description>The user source for Mozilla news, advocacy, interviews, builds, and more!
</description>
<language>en-us</language>  </channel>
```

The title can be treated as a headline link with the description following. The **Channel Language definition** allow aggregators to filter news feeds and gives the rendering software the information necessary to display the language properly. The </channnel> tag is used after all the channel elements to close the channel. As RSS conforms to XML specs, the element must be well formed; it requires the closing tag.

The image section

The image element is an optional element that is usually used to include the logo of the channel provider. The default size for the image is 88 pixels wide by 31 pixels high, but it can be enlarged to 144 pixels wide by 400 pixels wide.

Image element

```
<image><title>MozillaZine</title>
<url>http://www.mozillazine.org/image/mynetscape88.gif</url>
<link>http://www.mozillazine.org</link>
<width>88</width>
<height>31</height>  </image>
```

The image's title, URL, link, width, and height tags allow renderers to translate the file into HTML. The title tag is normally used for the image's ALT text.

The items

Items form the dynamic part of an RSS file. While channel, image, and text input elements create the channel's identity and typically stay the same over long periods of time, channel items are rendered as news headlines, and the channel's value depends on their changing fairly frequently.

Item element

```
<item><title>Java2 in Navigator 5?</title>
<link>http://www.mozillazine.org/talkback.html?article=607</link>
<description>Will Java2 be an integrated part of Navigator 5?
    Read more about it in this discussion...</description> </item>
```

Fifteen items are allowed in a channel. Titles should be less than 100 characters, while descriptions should be under 500 characters. The item title is normally rendered as a headline that links to the full article whose URL is provided by the item link. The item description is commonly used for either a summary of the article's content or for commentary on the article.

News feed channels use the description to highlight the content of news articles, usually on the channel owner's site, and Web log channels use the description to provide commentary on a variety of content, often on third-party sites.

The text input

The text input area is an optional element, with only one allowed per channel. This lets the user respond to the channel.

```
<textinput><title>Send</title>
<description>Comments about MozillaZine?</description>
<name>responseText</name>
<link>http://www.mozillazine.org/cgi-bin/sampleonly.cgi</link>    </textinput>
```

The title is normally rendered as the label of the form's ssubmit button, and the description as the text displayed before or above the input field. The text input name is supplied along with the contents of the text field when the submit button is clicked.

ATOM

Atom is a syndication data format like RSS, as well as a publishing protocol. The Atom data format uses XML syntax with one or more entry elements containing the full and/or summary content.

The **Atom Publishing Protocol** (APP) defines a hierarchy for organizing published content (services, workspaces, collections, and resources) and uses the HTTP Get, Post, Put, and Delete methods for retrieving, creating, deleting, and editing published content. Atom's use of HTTP's built-in methods and XML as a data format is in the spirit of a RESTful web services implementation.

Atom was designed to be a universal publishing standard for blogs and other Web sites where content is updated frequently. Users visiting a Web site with an Atom feed can discover a file described as "atom.xml" in the URL that can be copied and pasted into an aggregator to subscribe to the feed.

Atom was originally developed as an alternative to RSS 2.0, the standard developed by Dave Winer and copyrighted by Harvard University, as a means of improving perceived shortcomings of the RSS format by the blogging community.

Features of ATOM

- Atom was developed to be vendor neutral and freely extensible by any user; it is an open standard.
- Atom lies within an XML-namespace.
- Atom includes auto discovery, allowing feed aggregators to automatically detect the presence of a feed.
- Atom differentiates between relative and non-relative URIs.
- Atom has separate summary and content elements.
- Atom explicitly labels a payload as plaintext or HTML.
- Each entry has a globally unique ID.