MODULE-6
XML

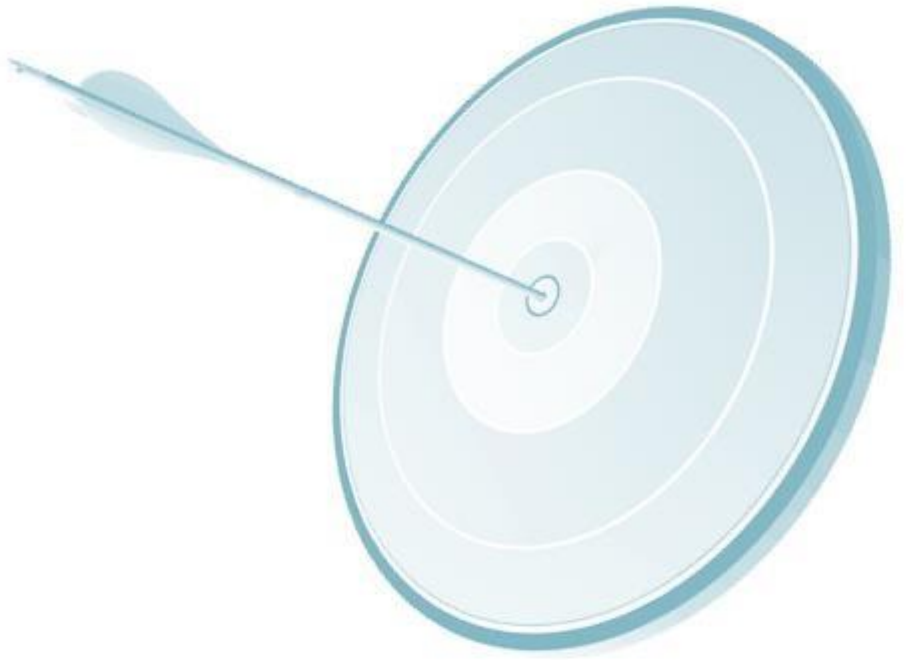
Course Topics

- → Module 1
  - » Introduction to Java
- → Module 2
  - » Data Handling and Functions
- → Module 3
  - » Object Oriented Programming in Java
- → Module 4
  - » Packages and Multi-threading
- → Module 5
  - » Collections
- → Module 6
  - » **XML**
- → Module 7
  - » JDBC
- → Module 8
  - » Servlets
- → Module 9
  - » JSP
- → Module 10
  - » Hibernate
- → Module 11
  - » Spring
- → Module 12
  - » Spring, Ajax and Design Patterns
- → Module 13
  - » SOA
- → Module 14
  - » Web Services and Project

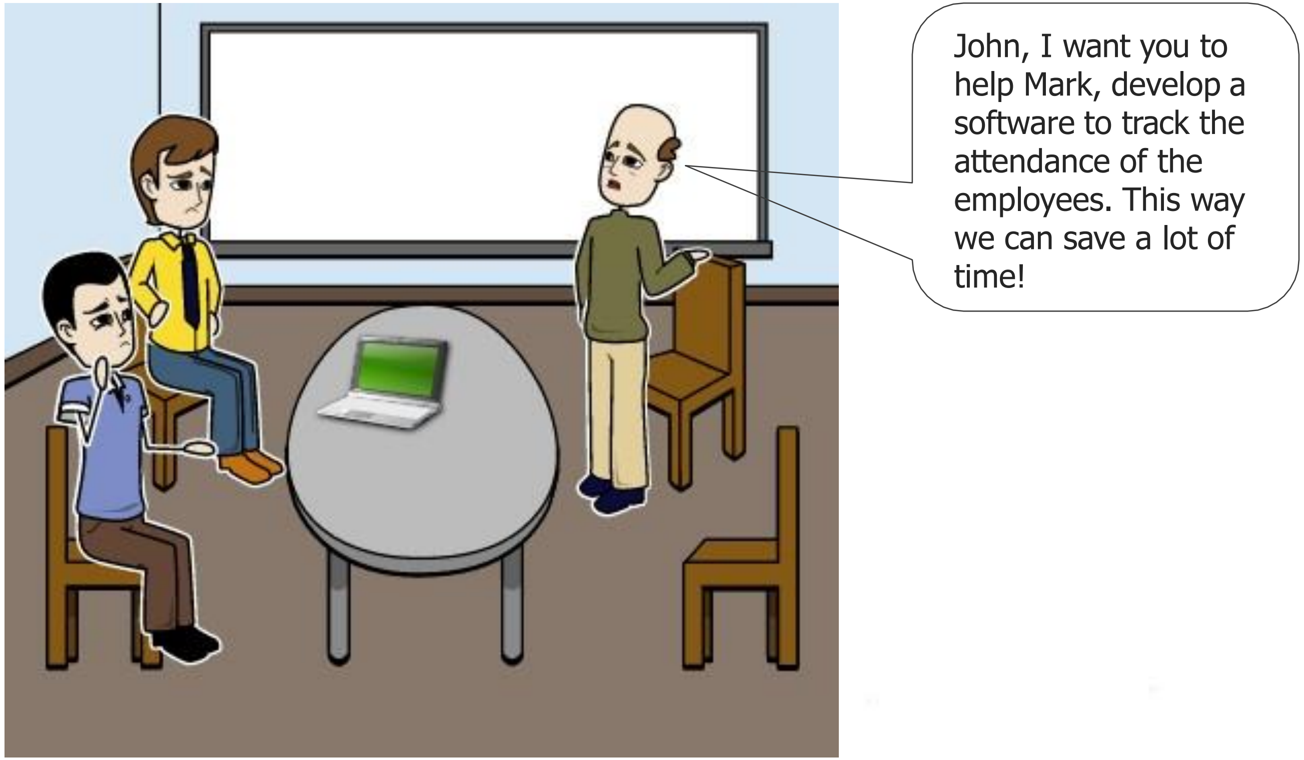
Objectives

At the end of this module, you will be able to

- → Understand why we need XML?
- → Understand the features of XML
- → Create and use XML files
- → Understand and use DTD
- → Understand XSD and XPath
- → Use XML parsers like SAX and DOM parser
- → Create and use XSL files

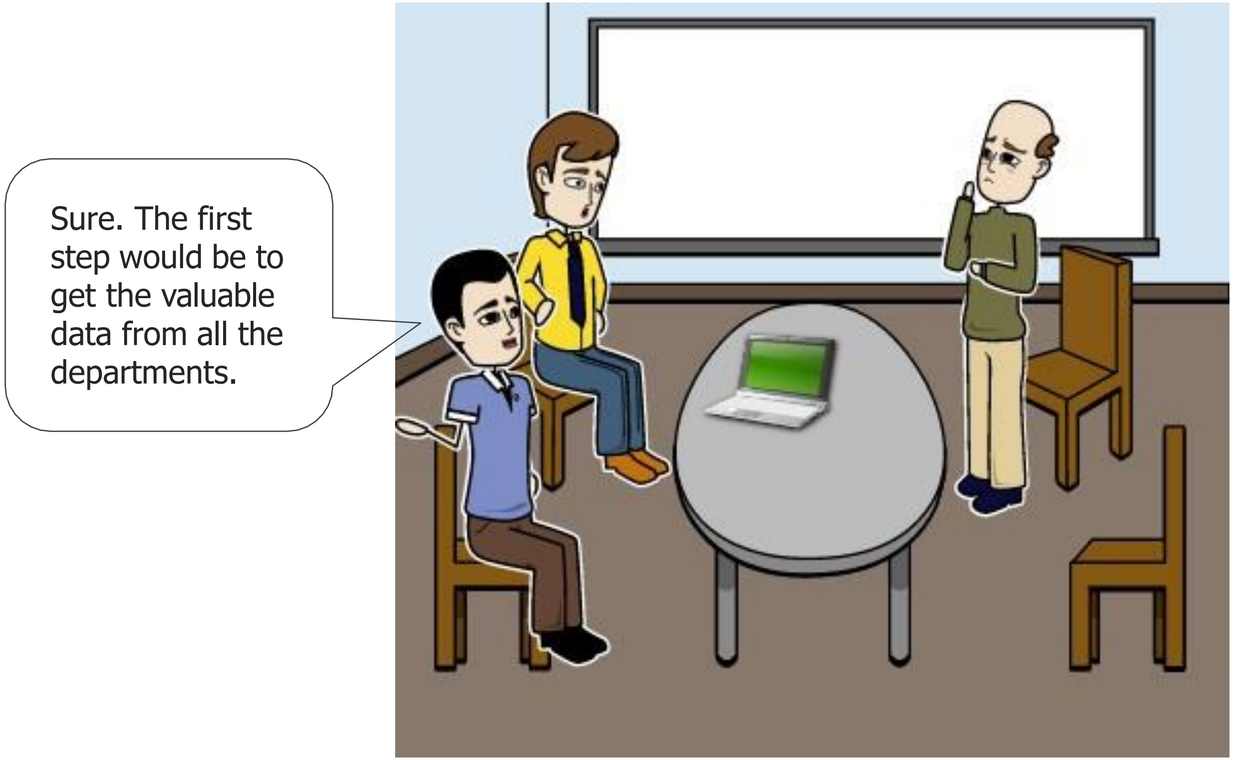


John works on a Project


John, I want you to help Mark, develop a software to track the attendance of the employees. This way we can save a lot of time!

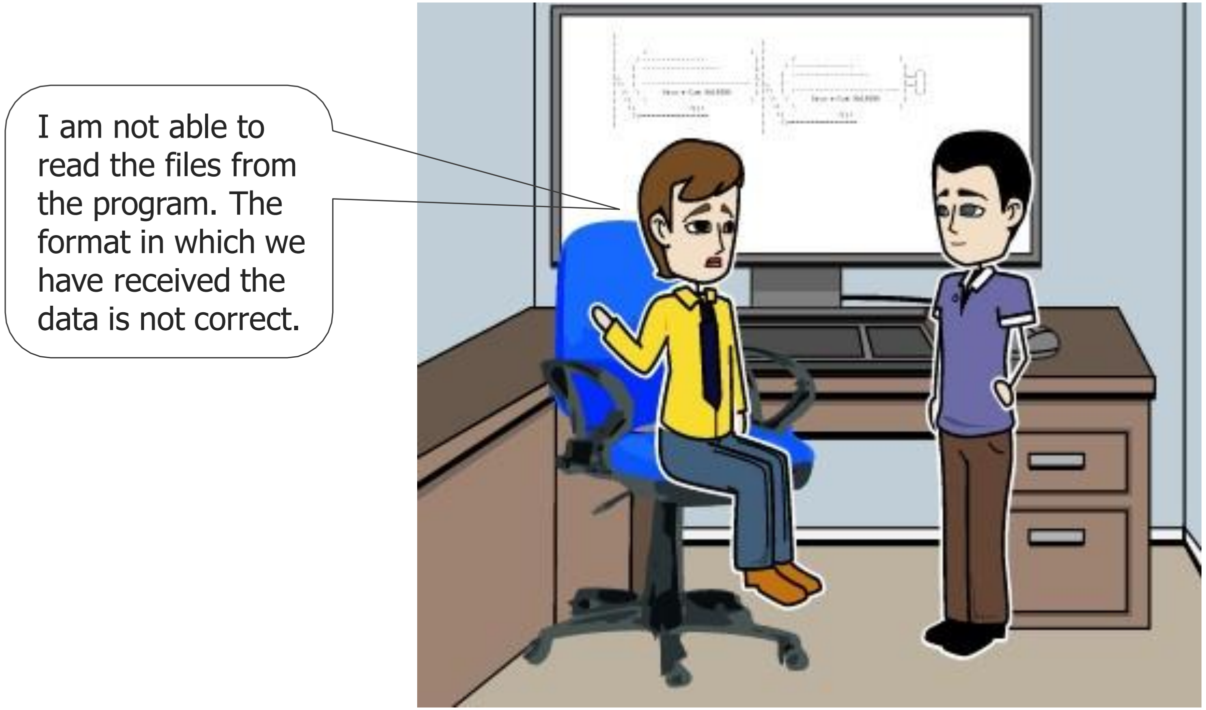
John Suggests the ways!

John instructs Mark to get the data from all departments….


Sure. The first step would be to get the valuable data from all the departments.

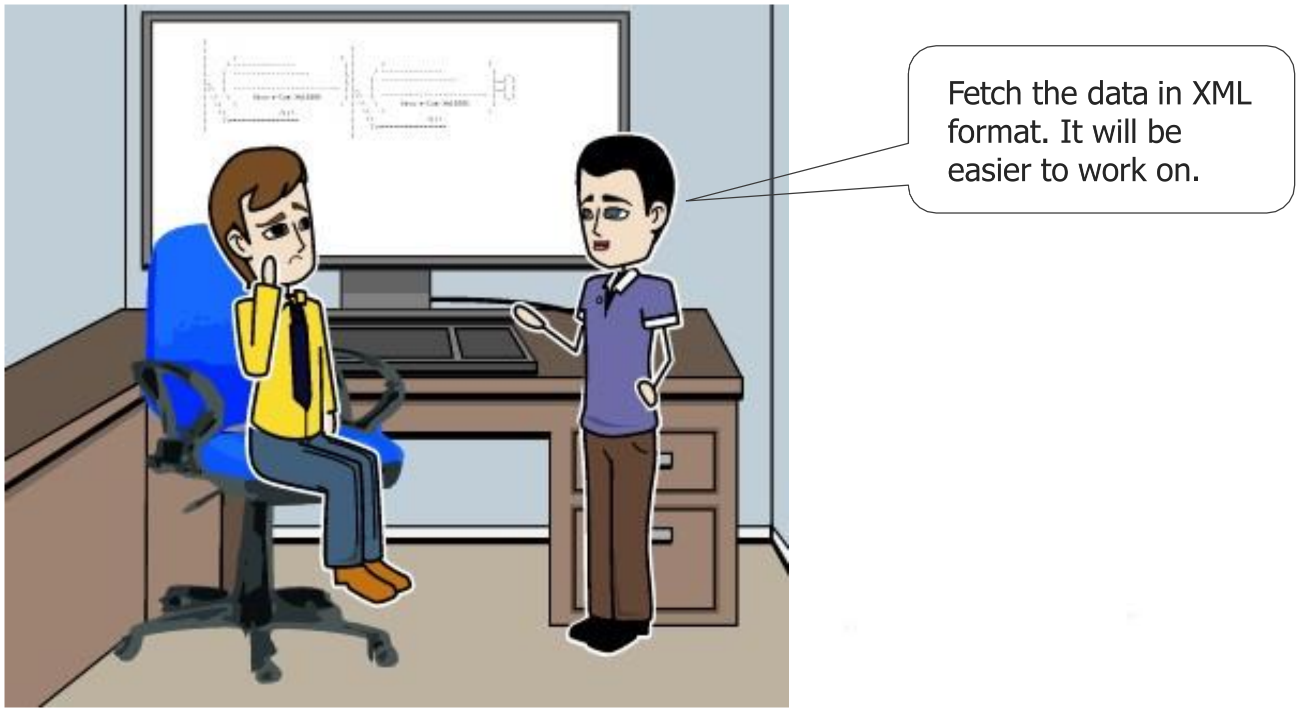
Mark Collects Data..

The organization has many departments like HR, Admin, R&D, Marketing, Sales, Technical Writing etc., Mark gets data from all departments in a different format. Mark is unable to read the files from the program as the format from each department is different.

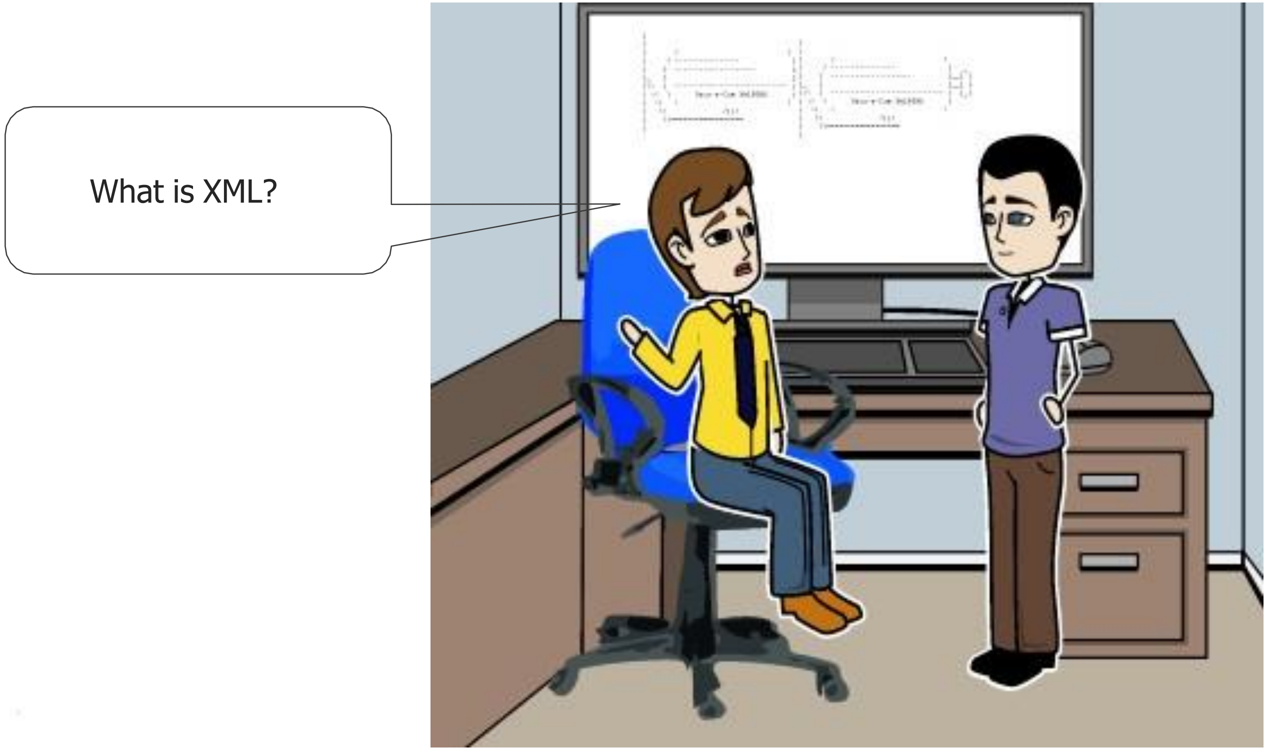

I am not able to read the files from the program. The format in which we have received the data is not correct.

Data in XML format

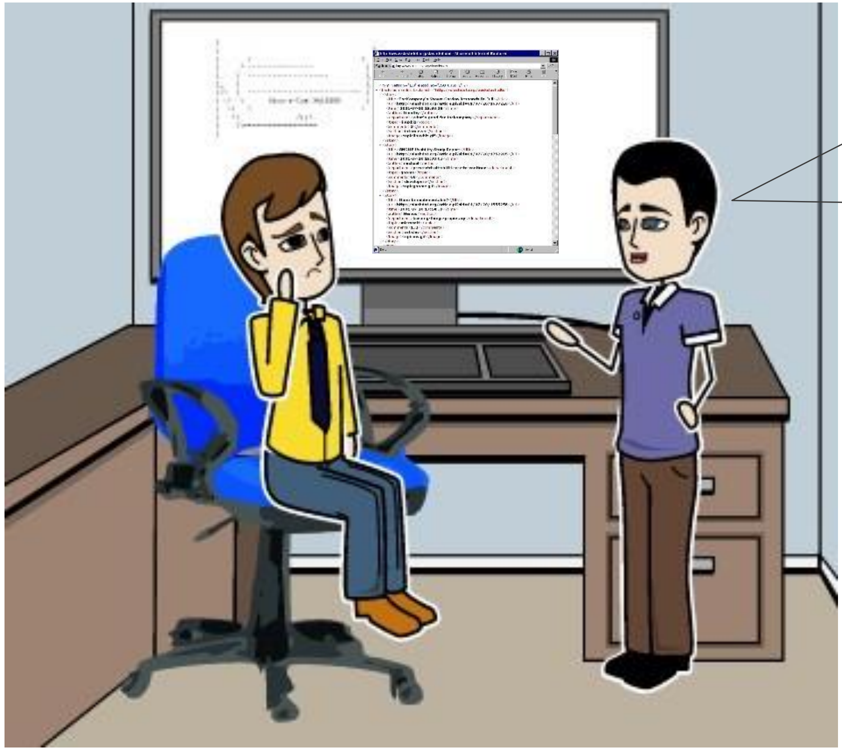
John instructs Mark to ask each department to give data in XML format by specifying the format for the data (DTD).



Mark has a doubt..

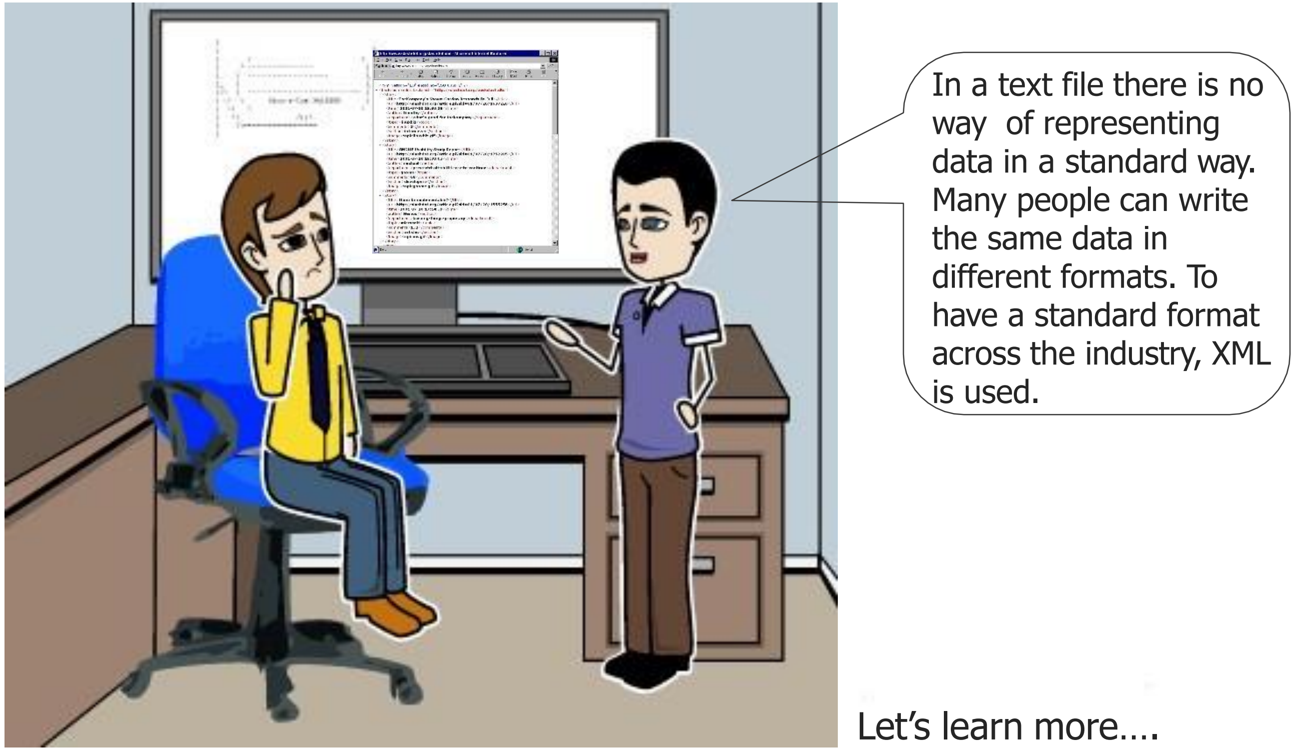


John explains XML



XML stands for Extensible Markup Language. XML enables you to work with more data from more sources, and to get more out of that data. It also helps in storing the data.

John explains XML


In a text file there is no way of representing data in a standard way. Many people can write the same data in different formats. To have a standard format across the industry, XML is used.

Let’s learn more….

Features of XML

Simplicity – Writing XML is very easy.

Extensibility – XML data can be extended with DTD and XSL. DTD is Document Type definition which provides the format of the XML document and XSL is the way XML data will be represented/displayed. It is like CSS for HTML.

Interoperability – It can work on any platform.

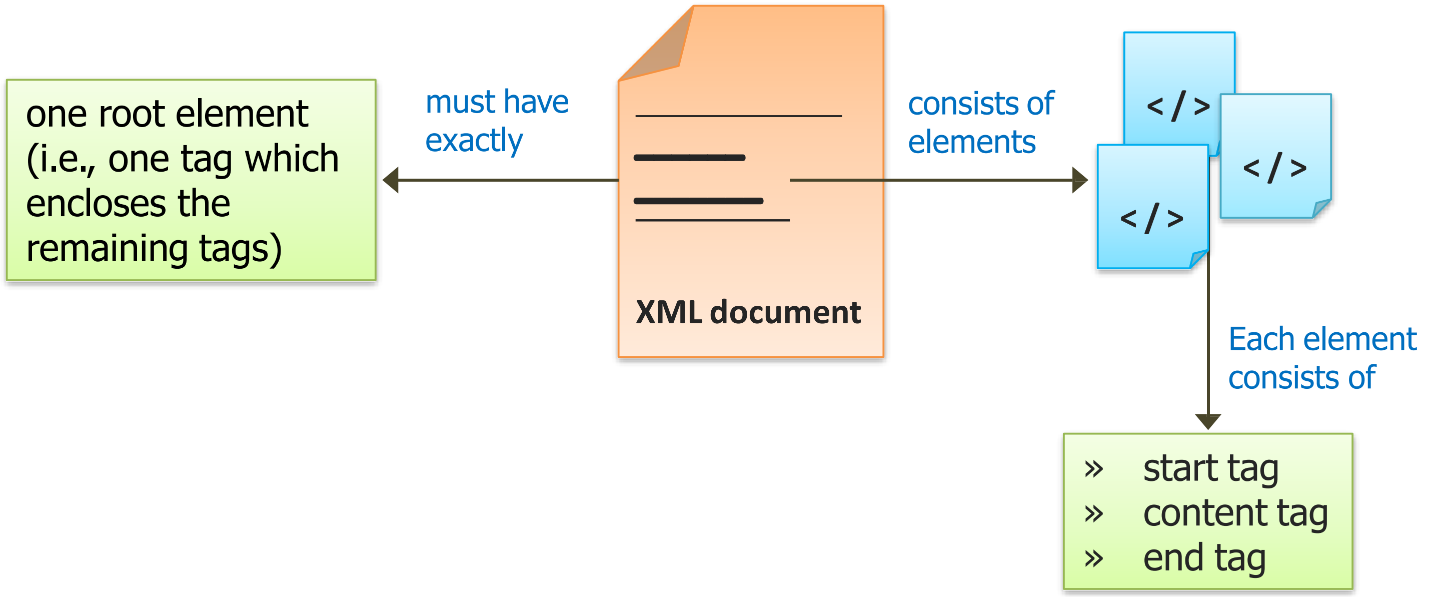
Openness – Any tool/language can open an xml file and parse it in the programming language.

XML

→ Data is inserted in the XML file in tags like HTML.

→ XML is written using user defined tags and data is written in between the user defined tags. User can open the xml file in any text editor or xml editors. It can also read/parsed using computer languages.

What does XML consists of?



Where can we use XML?

XML can be used to:

→ exchange data between different systems

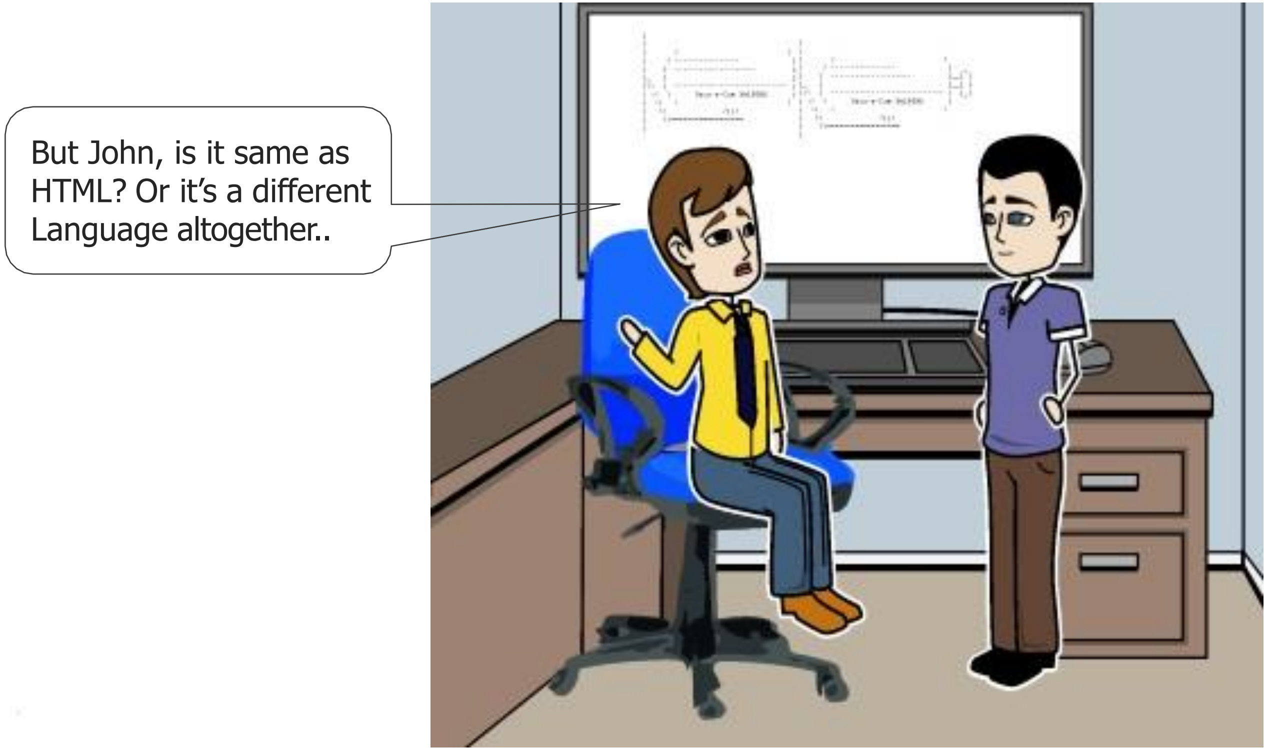
→ share data

→ store data

→ make data more useful

→ create new “ML Languages”

Is there a difference between XML and HTML?


But John, is it same as HTML? Or it’s a different Language altogether..

John explains the Difference..

HTML and XML differs in a lot of ways. I will explain a few….

Difference between HTML and XML

XML	HTML
XML is used for storing data and data communication.	HTML is used mainly used for display.
XML uses user-defined tags.	HTML uses built in tags.
XML is case sensitive.	HTML is case insensitive.
In XML it is mandatory to close all the tags.	In HTML its not mandatory to close the tags.

Simple XML File

XML file will have an extension of .xml for the file name.

When you double click on XML file, it can be opened in any browser.

Sample XML file is given below:

```
<person>
    <name> John </name>
    <class> Java </class>
    <profession> Trainer </profession>
    <location> California </location>
</person>
```

Simple XML File

Here <person>,<name>,<class>, <profession> and <address> are tags.

Data will be written in between these tags.

XML as Tree

XML forms a tree structure. Here root_element is the root of the tree, person is a branch and name, class, profession and location are the leaves of the tree.

```
<root_element>
      <person>
            <name> John </name>
            <class> Java </class>
            <profession> Trainer </profession>
            <location> California </location>
      </person>
      <person>
            <name> Amit </name>
            <class> Java </class>
            <profession> Software developer </profession>
            <location> California </location>
      </person>
</root_element>
```

XML Rules

→ White spaces are ignored in HTML but XML considers white spaces as the actual data.

→ Ordering and nesting of XML document should be proper. XML file does not work properly if there is clash in nesting.

→ XML tags are case sensitive. Opening and closing tags should be exactly the same without any difference in the case.

→ Every opening tag must have a close tag else XML file will not work correctly.

XML Namespace

In XHTML, HTML and XML are valid.

```
<html>
        <body>
                <p>Patient Information</p>
        </body>
        <body>
                <height>5.6ft</height>
                <weight>75kgs</weight>
        </body>
</html>
```

Here body is used for two difference purposes. 1 is HTML body and another one is patient's body information. To resolve this issue, name spaces are used.

XML Namespace

```
html:html
xmlns:html='http://www.w3.org/xhtml1/'>
<html:body>
<html:h4>Patient Information</html:h4>
</HTML:body>
<health:body
xmlns:health='http://www.test.com/health'
>
<health:height>5.6ft</height>
<health:weight>75kgs</weight>
</health:body>
</html:html>
```

Here if the html and health namespace is used. If html body means it is html and if it health body means patient's body information. To resolve the conflict, name spaces are used.

XML Names Spaces

Syntax of Name space is:

```
Element:name space xmlns:name space=”name space identifier”
```

XML Prolog

First line of XML is called Prolog.

Prolog will contain the following:

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

→ Prolog usually have version number and encoding.

→ Encoding is like character set. This is used for Unicode.

→ UTF stands for Universal character set Transformation Format.

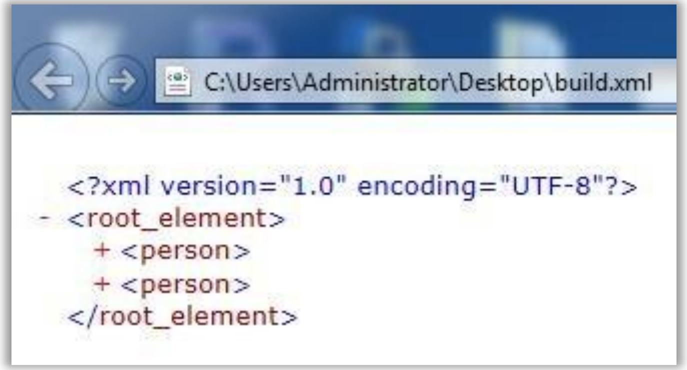
→ UTF-8 is used for standard web applications.

Editing and Viewing the XML Files

Editing the XML file can done in any regular text editor or you can use specific XML editors.

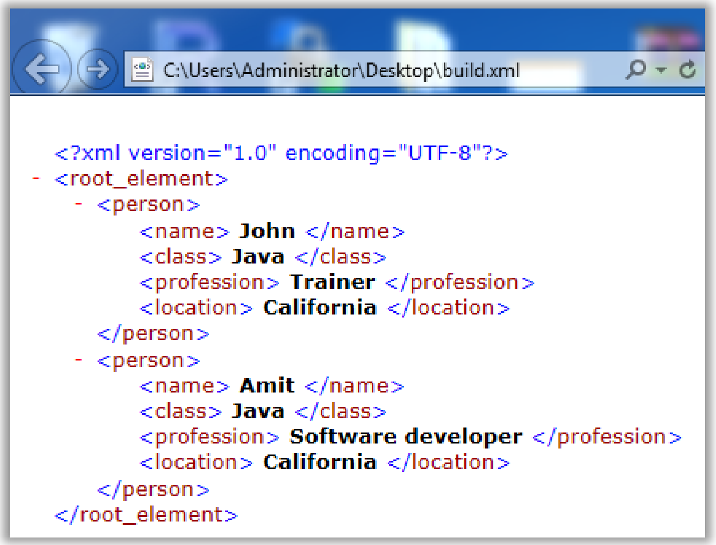
Viewing the xml file can be seen in the browser too.

When an XML file is opened in the browser, + and – marks also shown. + means this tag is a branch and leaves exist below this tag. - means leaf element of the xml file.



C:\Users\Administrator\Desktop\build.xml

```
  <?xml version="1.0" encoding="UTF-8"?>
- <root_element>
    + <person>
    + <person>
  </root_element>
```



C:\Users\Administrator\Desktop\build.xml

```
  <?xml version="1.0" encoding="UTF-8"?>
- <root_element>
  - <person>
        <name> John </name>
        <class> Java </class>
        <profession> Trainer </profession>
        <location> California </location>
    </person>
  - <person>
        <name> Amit </name>
        <class> Java </class>
        <profession> Software developer </profession>
        <location> California </location>
    </person>
  </root_element>
```

LAB

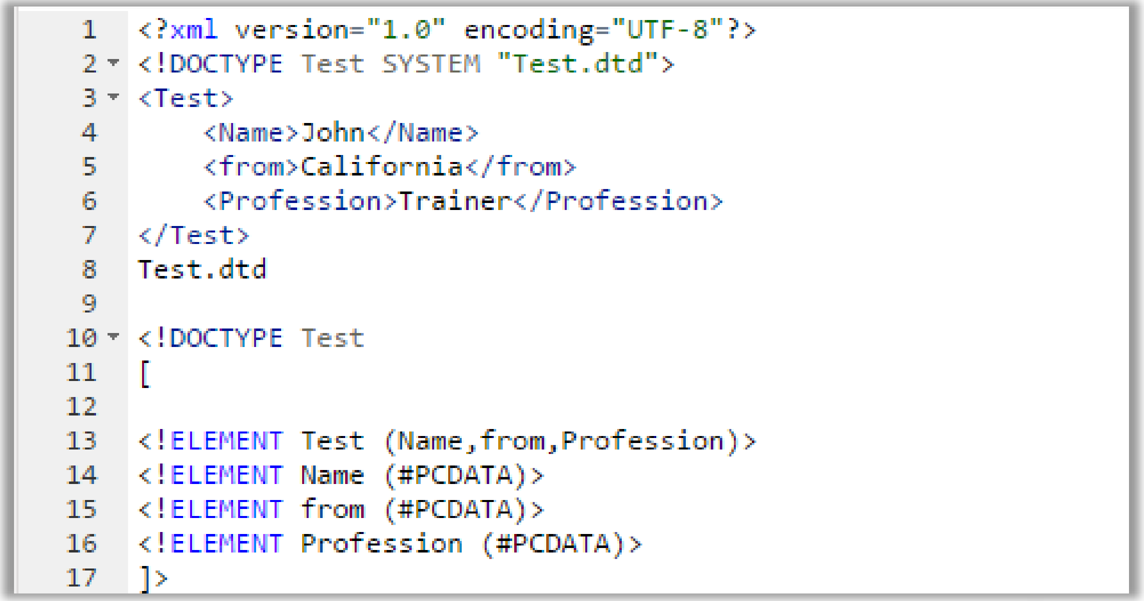
XML DTD

→ DTD stands for Document Type Definition.

→ DTD specifies the format of the XML Document.

→ DTD can be inside an XML file or outside the XML file.

DTD



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Test SYSTEM "Test.dtd">
<Test>
    <Name>John</Name>
    <from>California</from>
    <Profession>Trainer</Profession>
</Test>
Test.dtd

<!DOCTYPE Test
[

<!ELEMENT Test (Name,from,Profession)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT Profession (#PCDATA)>
]>
```

DTD

→ DOCTYPE says the root element of the XML file.

→ ELEMENT Test says it has 3 elements which are Name, from and Profession.

→ All the ELEMENTS are PCDATA, that means all the elements are parsable character data.

→ At times you need not parse certain elements then you can use CDATA.

→ Using XML parser, using validate() we can validate XML whether it is formed according to DTD or not.

Why DTD?

→ Even the XML file can be written in different formats by different people.

→ To avoid this issue and to standardize the XML data in one format DTD is used.

Features of DTD

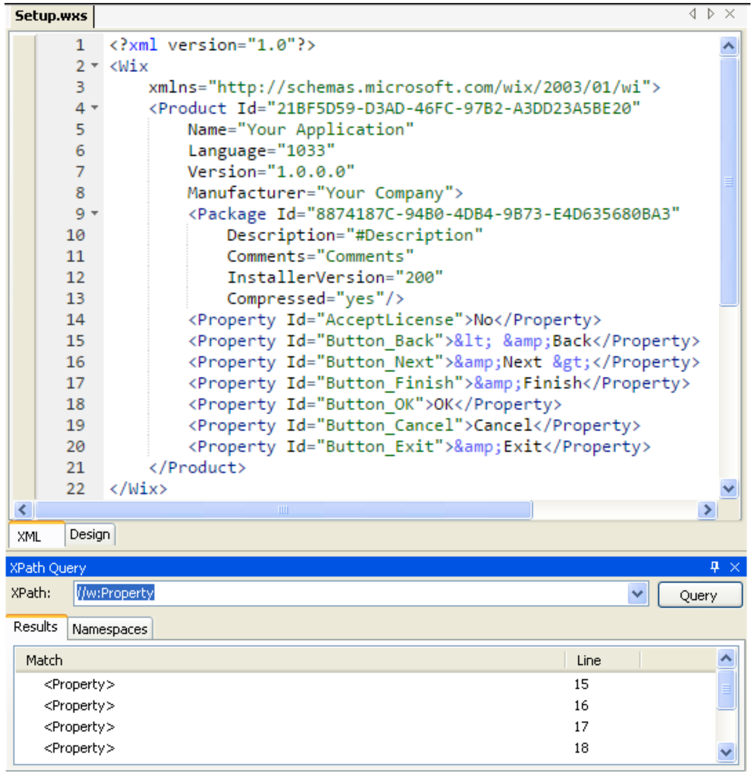
→ Providing standard way of representing data.

→ XML data can be validated with DTD.

→ DTD can be given in the xml file or it can be given in a separate file having extension of .dtd and can use it in the XML file.

→ During the validation of XML, parsers read the DTD file and validates against the XML file. Finally validator will give the result as valid XML or Invalid XML.

XPath



Setup.wxs

```
 1  <?xml version="1.0"?>
 2  <Wix
 3      xmlns="http://schemas.microsoft.com/wix/2003/01/wi">
 4      <Product Id="21BF5D59-D3AD-46FC-97B2-A3DD23A5BE20"
 5          Name="Your Application"
 6          Language="1033"
 7          Version="1.0.0.0"
 8          Manufacturer="Your Company">
 9          <Package Id="8874187C-94B0-4DB4-9B73-E4D635680BA3"
10              Description="#Description"
11              Comments="Comments"
12              InstallerVersion="200"
13              Compressed="yes"/>
14          <Property Id="AcceptLicense">No</Property>
15          <Property Id="Button_Back">&lt; &amp;Back</Property>
16          <Property Id="Button_Next">&amp;Next &gt;</Property>
17          <Property Id="Button_Finish">&amp;Finish</Property>
18          <Property Id="Button_OK">OK</Property>
19          <Property Id="Button_Cancel">Cancel</Property>
20          <Property Id="Button_Exit">&amp;Exit</Property>
21      </Product>
22  </Wix>
```

XPath Query

XPath: //w:Property

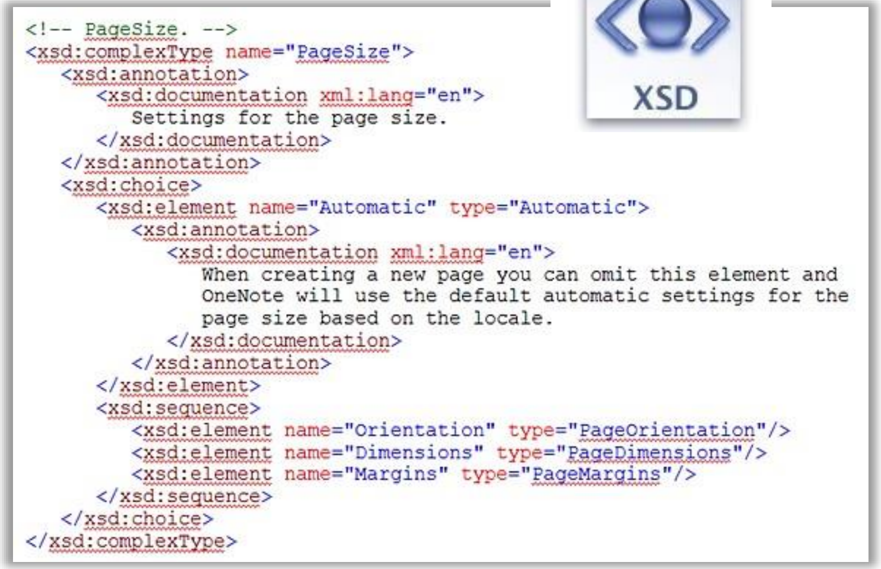
Match	Line
<Property>	15
<Property>	16
<Property>	17
<Property>	18

→ XPath is a language for addressing parts of an XML document, designed to be used by both XSLT(XSL Transformations) and XPointer to provide a common syntax and semantics for functionality shared between them.

→ The primary purpose of XPath is to address parts of an XML [XML] document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans.

http://community.sharpdevelop.net/blogs/mattward/archive/2006/08/05/TestingXPathQueriesInSharpDevelop.aspx

XSD (XML Schema Definition)



```
<!-- PageSize. -->
<xsd:complexType name="PageSize">
   <xsd:annotation>
      <xsd:documentation xml:lang="en">
         Settings for the page size.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:choice>
      <xsd:element name="Automatic" type="Automatic">
         <xsd:annotation>
            <xsd:documentation xml:lang="en">
               When creating a new page you can omit this element and
               OneNote will use the default automatic settings for the
               page size based on the locale.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:sequence>
         <xsd:element name="Orientation" type="PageOrientation"/>
         <xsd:element name="Dimensions" type="PageDimensions"/>
         <xsd:element name="Margins" type="PageMargins"/>
      </xsd:sequence>
   </xsd:choice>
</xsd:complexType>
```

→ An XML schema represents the interrelationship between the attributes and elements of an XML object (for example, a document or a portion of a document).

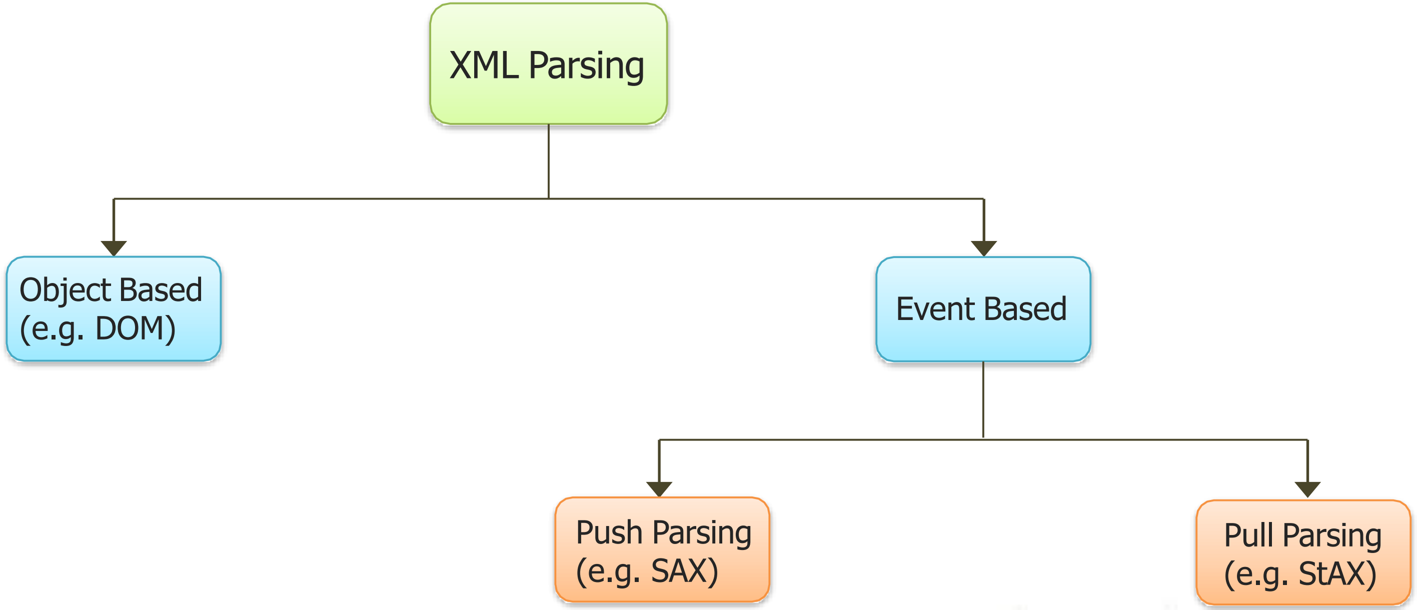
→ This description can be used to verify that each item of content in a document adheres to the description of the element in which the content is to be placed.

→ To create a schema for a document, you analyse its structure, defining each structural element as you encounter it.

http://blogs.msdn.com/b/johnguin/archive/2009/06/02/looking-at-the-xml-schema-for-page-width-and-height-in-onenote.aspx

XML Parsers

Reading XML file is called parsing.



Difference between DOM and SAX Parsers

DOM Parsers	SAX Parsers
DOM stands for Document Object Model	SAX stands for Simple API for XML
DOM reads the entire document	SAX reads node by node
DOM is useful when reading small to medium size XML files	SAX is used when big XML files needs to be parsed
DOM is tree based parser	SAX is event based parser
DOM is little slow as compared to SAX	SAX is faster than DOM
DOM can insert and delete nodes	SAX cannot insert and delete nodes

Why DOM and SAX?

From the program, if the xml file data is to be read then developer has to write their own code to parse the xml file. It is complicated.

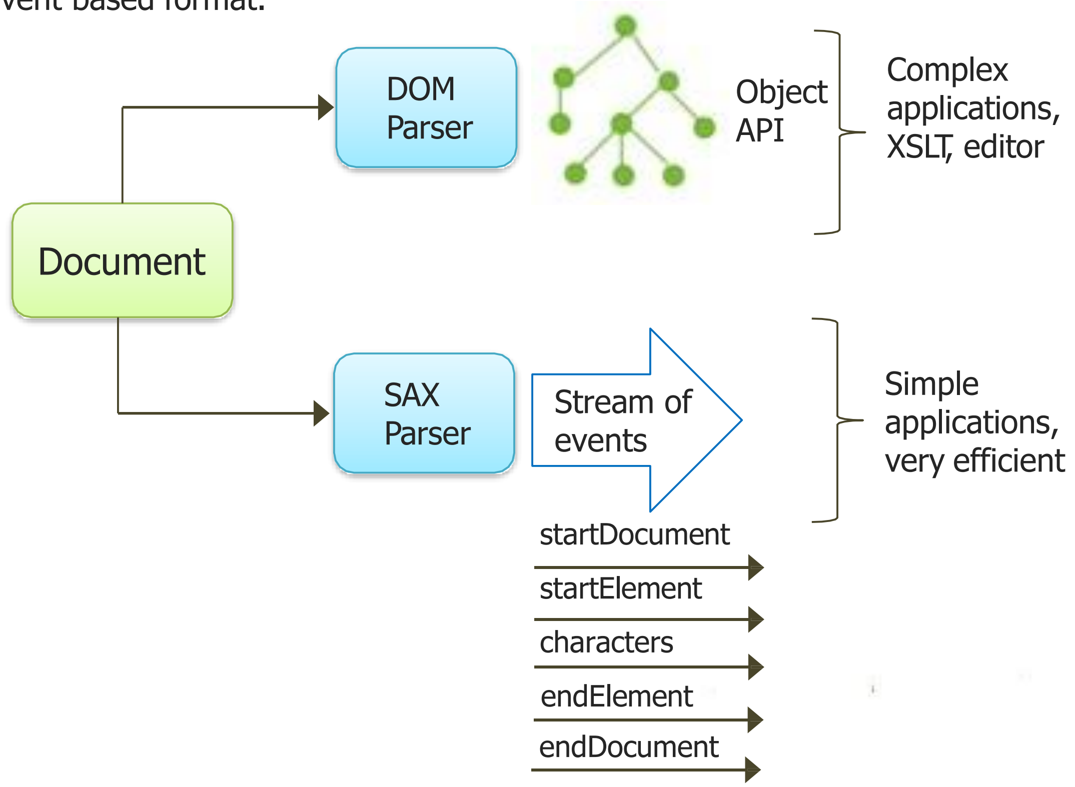
To avoid this, Java/programming language has given parsers like DOM and SAX to read the xml file.

XML parser consumes lot of time and it is complicated, DOM/SAX parser is used to parse the xml file. Using this, the code can be developed fast and accurate.

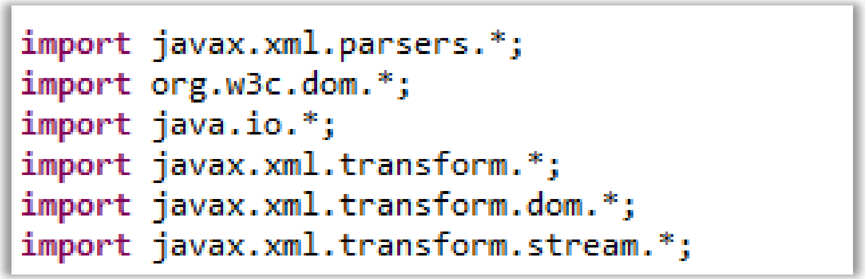
Features of DOM and SAX?

→ DOM reads the XML file and stores in tree format.

→ SAX reads in event based format.

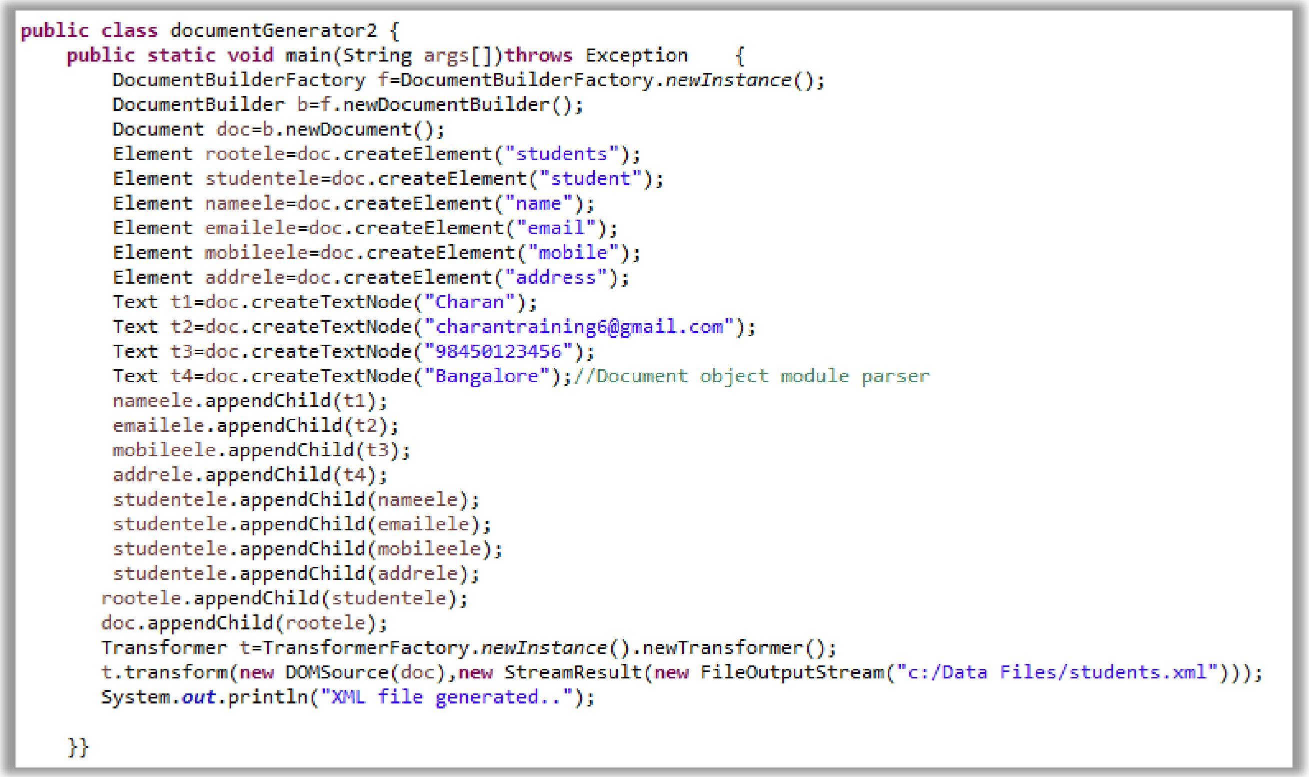


DOM parsing – creating XML file using DOM – Required Import



```java
import javax.xml.parsers.*;
import org.w3c.dom.*;
import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
```

DOM to Create XML file



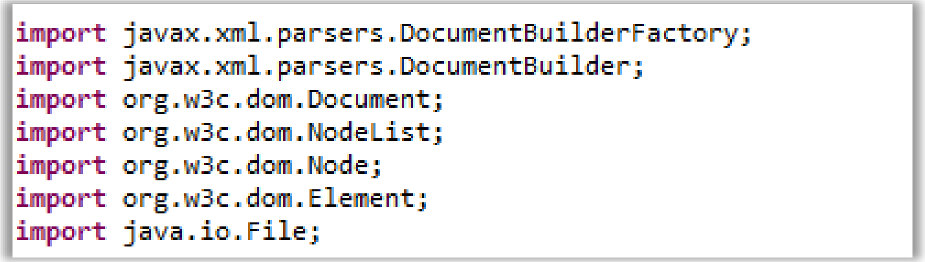
```
public class documentGenerator2 {
    public static void main(String args[])throws Exception    {
        DocumentBuilderFactory f=DocumentBuilderFactory.newInstance();
        DocumentBuilder b=f.newDocumentBuilder();
        Document doc=b.newDocument();
        Element rootele=doc.createElement("students");
        Element studentele=doc.createElement("student");
        Element nameele=doc.createElement("name");
        Element emailele=doc.createElement("email");
        Element mobileele=doc.createElement("mobile");
        Element addrele=doc.createElement("address");
        Text t1=doc.createTextNode("Charan");
        Text t2=doc.createTextNode("charantraining6@gmail.com");
        Text t3=doc.createTextNode("98450123456");
        Text t4=doc.createTextNode("Bangalore");//Document object module parser
        nameele.appendChild(t1);
        emailele.appendChild(t2);
        mobileele.appendChild(t3);
        addrele.appendChild(t4);
        studentele.appendChild(nameele);
        studentele.appendChild(emailele);
        studentele.appendChild(mobileele);
        studentele.appendChild(addrele);
       rootele.appendChild(studentele);
       doc.appendChild(rootele);
       Transformer t=TransformerFactory.newInstance().newTransformer();
       t.transform(new DOMSource(doc),new StreamResult(new FileOutputStream("c:/Data Files/students.xml")));
       System.out.println("XML file generated..");

    }}
```

DOM to Create an XML file

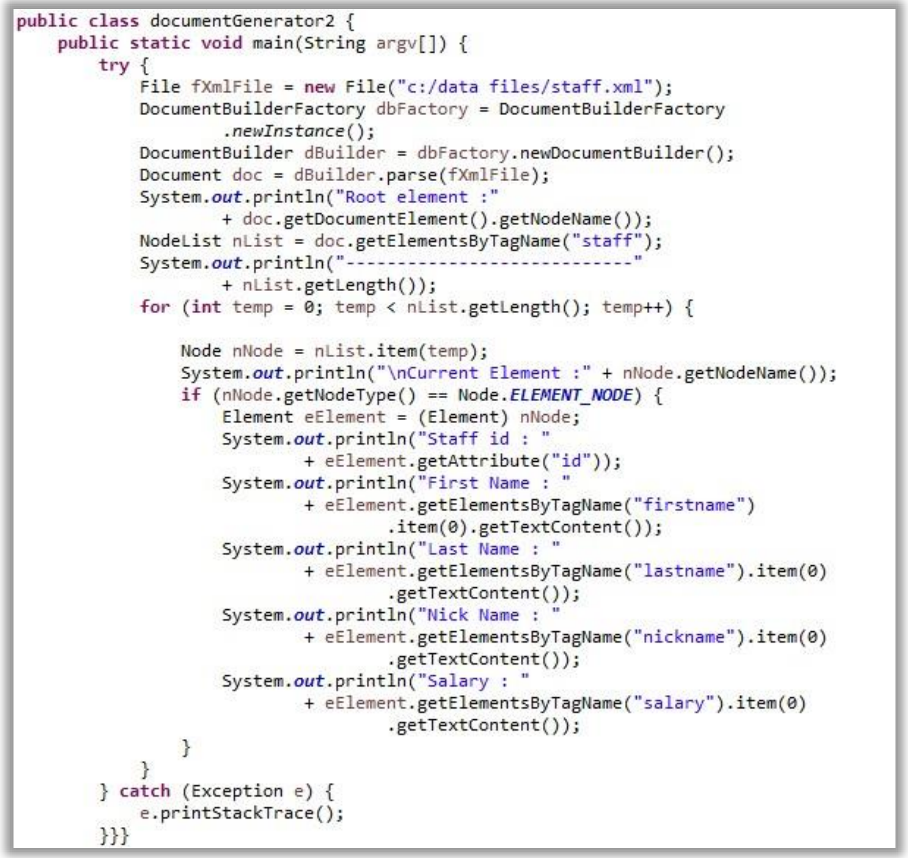
- → In the program, document object is created.
- → Elements/nodes of the XML file are created.
- → Data to be placed in between the nodes are created in text object.
- → Text and elements (nodes) are linked.
- → Nodes are linked to the Root node.
- → Finally, transformer Transform() method is used to convert the document object to XML file.

Parsing XML file using DOM – Required Imports



```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;
import java.io.File;
```

Parsing XML file using DOM



```java
public class documentGenerator2 {
    public static void main(String argv[]) {
        try {
            File fXmlFile = new File("c:/data files/staff.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory
                    .newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(fXmlFile);
            System.out.println("Root element :"
                    + doc.getDocumentElement().getNodeName());
            NodeList nList = doc.getElementsByTagName("staff");
            System.out.println("----------------------------"
                    + nList.getLength());
            for (int temp = 0; temp < nList.getLength(); temp++) {

                Node nNode = nList.item(temp);
                System.out.println("\nCurrent Element :" + nNode.getNodeName());
                if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element eElement = (Element) nNode;
                    System.out.println("Staff id : "
                            + eElement.getAttribute("id"));
                    System.out.println("First Name : "
                            + eElement.getElementsByTagName("firstname")
                                    .item(0).getTextContent());
                    System.out.println("Last Name : "
                            + eElement.getElementsByTagName("lastname").item(0)
                                    .getTextContent());
                    System.out.println("Nick Name : "
                            + eElement.getElementsByTagName("nickname").item(0)
                                    .getTextContent());
                    System.out.println("Salary : "
                            + eElement.getElementsByTagName("salary").item(0)
                                    .getTextContent());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }}}
```

Parsing XML file using DOM

XML file is read into tree using DOM parse()

Number of branches are determined and all the leaf nodes are read through iterating all the branches.

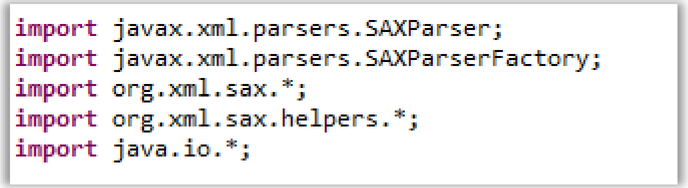
In DOM,

GetNodeName() → Returns the name of the current node.

GetAttribute() → Returns the value of XML Attribute node.

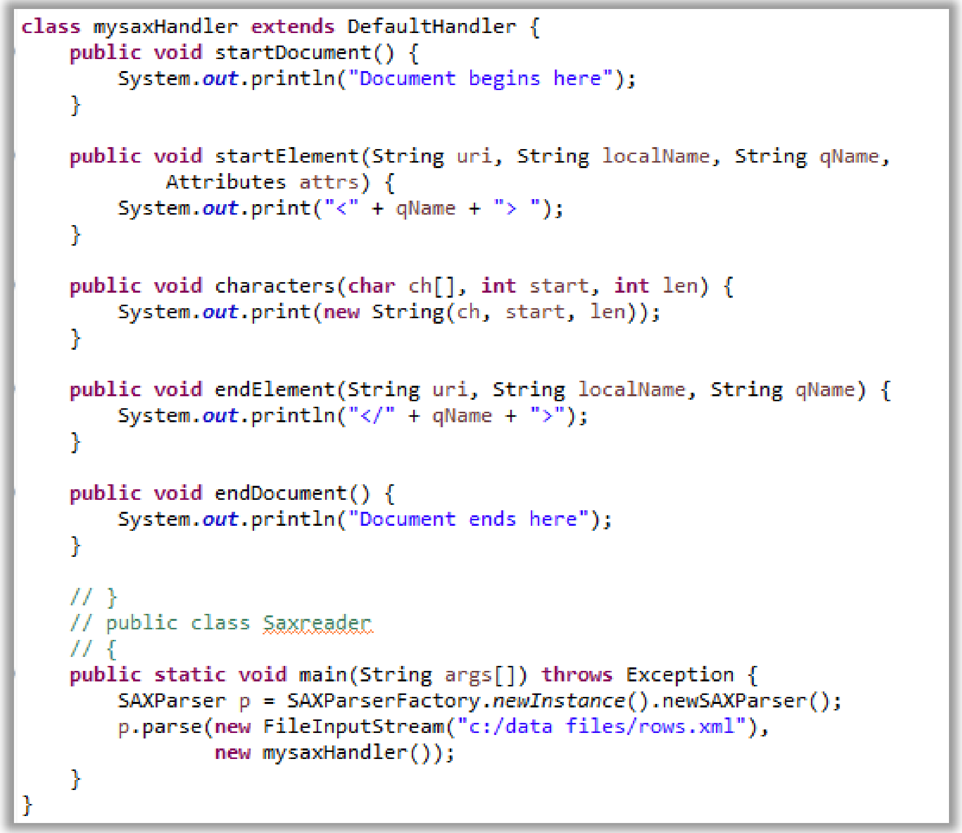
GetElementByTagName() → Returns the data of the node for the corresponding node name.

SAX Parser – Required Imports



```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;
```

SAX Parser



```
class mysaxHandler extends DefaultHandler {
    public void startDocument() {
        System.out.println("Document begins here");
    }

    public void startElement(String uri, String localName, String qName,
            Attributes attrs) {
        System.out.print("<" + qName + "> ");
    }

    public void characters(char ch[], int start, int len) {
        System.out.print(new String(ch, start, len));
    }

    public void endElement(String uri, String localName, String qName) {
        System.out.println("</" + qName + ">");
    }

    public void endDocument() {
        System.out.println("Document ends here");
    }

    // }
    // public class Saxreader
    // {
    public static void main(String args[]) throws Exception {
        SAXParser p = SAXParserFactory.newInstance().newSAXParser();
        p.parse(new FileInputStream("c:/data files/rows.xml"),
                new mysaxHandler());
    }
}
```

SAX Parser

→ In the main () method, XML file is parsed by SAXParser object and is attached with a mySaxHandler class.

→ For every node or data read, it calls a corresponding method in mysaxHandler class.

→ StartDocument and endDocument() methods are called when an XML document is started to read and after reading the entire document.

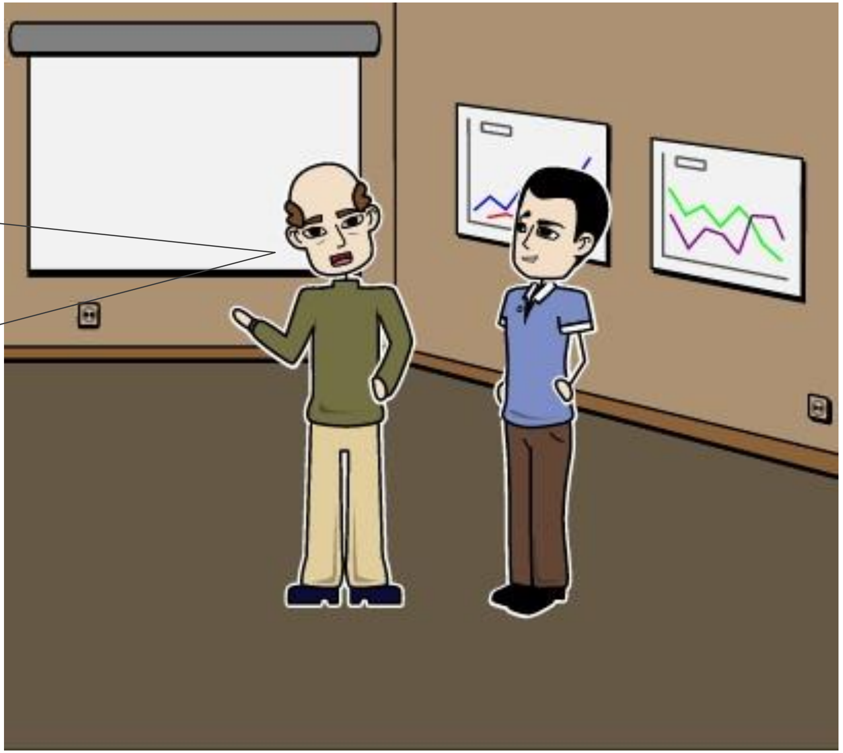
→ When start tag is read, startElement() method is invoked.

→ When end tag is read, endElement() method is invoked.

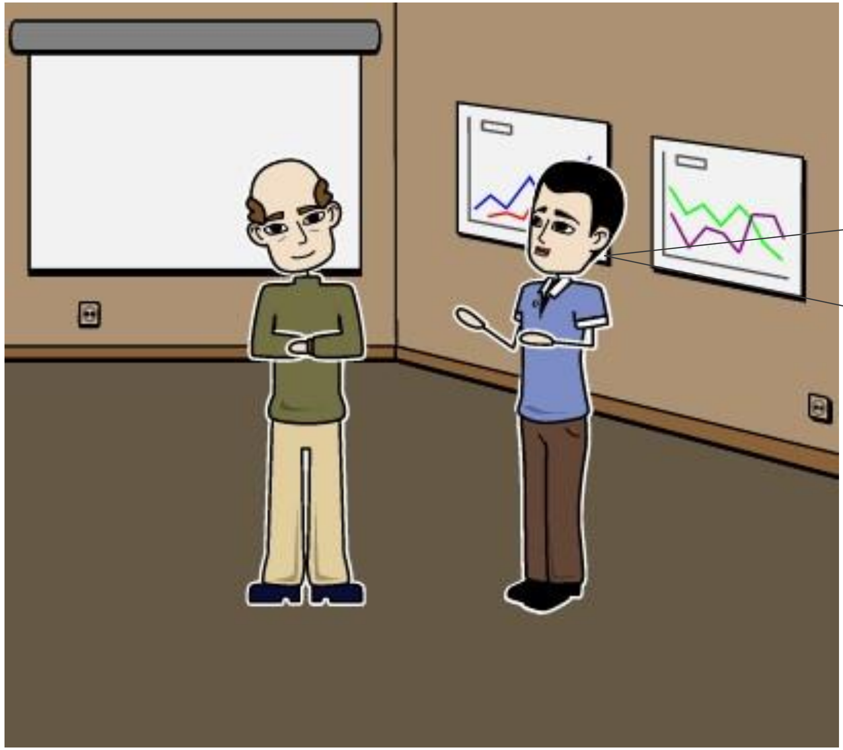
→ When the data in-between the tag is read, characters() method is called.

John works on a new Project..

John, we have a Project. I have an XML page that describes the characteristics of automobiles for an insurance company, (a set of open and close tags might contain the name of an auto manufacturer). What can I do to display the name in the web browser?



John works on a new Project..



Sir, this can be done using XSL -Extensible Stylesheet Language. Using XSL, we can tell the web browser to display the manufacturer’s name. Not only this we can fix the place holder and the font too.

Why XSL?

→ XML is used only to store data.

→ We need a mechanism to display the data in a specified format. This is possible using XSL.

What is XSL?

→ XSL stands for Extensible Stylesheet Language.

→ Like the way CSS is stylesheet for HTML, XSL is the stylesheet for XML.

→ XSL can navigate all the nodes/elements of the XML file and can display the XML data in the particular format. This is done by XSLT which stands for Extensible stylesheet Language Transformations.

Features of XSL

→ Representing the XML data in the required format.

→ Queries also can be specified in XSL.

→ XSL file reads the XML data, looks for the query (If it is given in the XSL file) and displays the data on browser in the format given in the XSL file.

XSL File

Browser should support XSLT then other wise the above code will not work.

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

→ This line should be used in the XML file to use the XSL file.

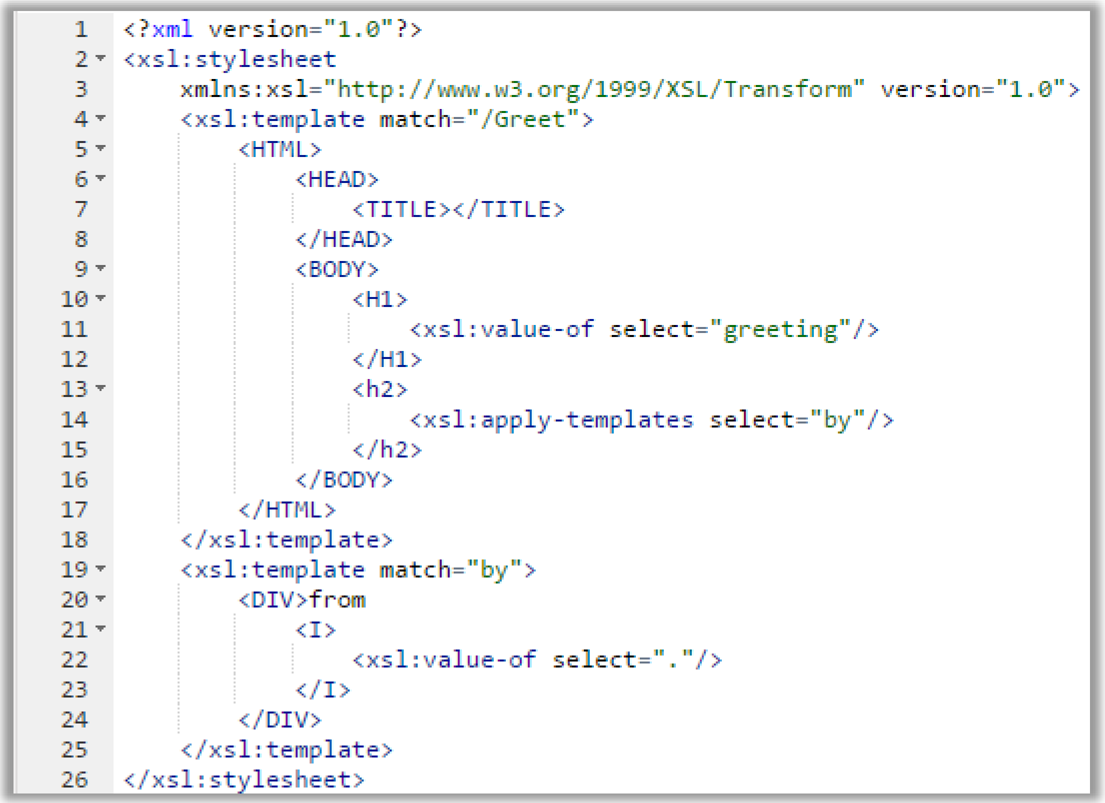
→ This line will use the XML data and use XSL format and displays XHTML in the browser.

LAB

Sample XML File

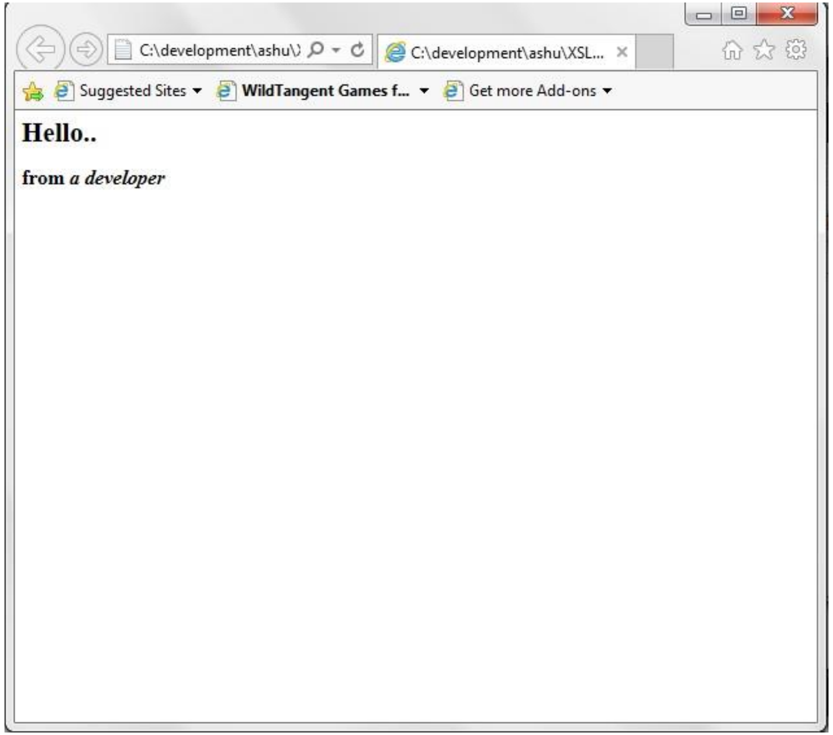
```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl"
href="Welcome.xsl"?>
<Greet>
  <by>a developer</by>
  <greeting>Hello..</greeting>
</Greet>
```

Sample XSL File



```
<?xml version="1.0"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="/Greet">
        <HTML>
            <HEAD>
                <TITLE></TITLE>
            </HEAD>
            <BODY>
                <H1>
                    <xsl:value-of select="greeting"/>
                </H1>
                <h2>
                    <xsl:apply-templates select="by"/>
                </h2>
            </BODY>
        </HTML>
    </xsl:template>
    <xsl:template match="by">
        <DIV>from
            <I>
                <xsl:value-of select="."/>
            </I>
        </DIV>
    </xsl:template>
</xsl:stylesheet>
```

XML file will display



Hello..

from *a developer*

Assignment: XML File

→ Write an XML file student having the information about Student's id, name, std, gender, marks and address.

Assignment: DOM and SAX Parser

→ Use the xml file created by you in the previous assignment using DOM and parse it using SAX and DOM parsers and display the content of the XML.

Assignment: DOM

→ Write a XML file containing Employee details in the XML File using DOM. Fields of the XML are:

- » Emp_id
- » name
- » Dept

Agenda of the Next Class

In the next module, you will be able to:

→ Understand database

→ Understand RDBMS and its Advantages

→ Execute SQL Queries

→ Create JDBC connection to the database

→ Perform CRUD operation on the database from your Java application

→ Perform Batch processing and Transaction Management

Pre-work

- Read Database concepts and SQL for better understanding of the topics in the next class.

QUESTIONS

Thank you!