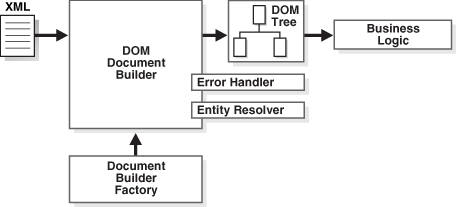
For example purpose, We will be parsing below xml content in all code examples.

|  |
| --- |
| <employees>      <employee id="111">          <firstName>Lokesh</firstName>          <lastName>Gupta</lastName>          <location>India</location>      </employee>      <employee id="222">          <firstName>Alex</firstName>          <lastName>Gussin</lastName>          <location>Russia</location>      </employee>      <employee id="333">          <firstName>David</firstName>          <lastName>Feezor</lastName>          <location>USA</location>      </employee>  </employees> |

**1. DOM Parser API**

Let’s note down some broad steps to **create and use DOM parser** to parse a XML file in java.

DOM Parser in Action

**1.1. Import dom parser packages**

We will need to import dom parser packages first in our application.

|  |
| --- |
| import org.w3c.dom.\*;  import javax.xml.parsers.\*;  import java.io.\*; |

**1.2. Create DocumentBuilder**

Next step is to create the DocumentBuilder object.

|  |
| --- |
| DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();  DocumentBuilder builder = factory.newDocumentBuilder(); |

**1.3. Create Document object from xml file**

Read the XML file to Document object.

|  |
| --- |
| Document document = builder.parse(new File( file )); |

**1.4. Validate Document structure**

XML validation is optional but good to have it before start parsing.

|  |
| --- |
| Schema schema = null;  try {    String language = XMLConstants.W3C\_XML\_SCHEMA\_NS\_URI;    SchemaFactory factory = SchemaFactory.newInstance(language);    schema = factory.newSchema(new File(name));  } catch (Exception e) {      e.printStackStrace();  }  Validator validator = schema.newValidator();  validator.validate(new DOMSource(document)); |

**1.5. Extract the root element**

We can get the root element from XML document using below code.

|  |
| --- |
| Element root = document.getDocumentElement(); |

**1.6. Examine attributes**

We can examine the xml element attributes using below methods.

|  |
| --- |
| element.getAttribute("attributeName") ;    //returns specific attribute  element.getAttributes();                //returns a Map (table) of names/values |

**1.7. Examine sub-elements**

Child elements can inquired in below manner.

|  |
| --- |
| node.getElementsByTagName("subElementName") //returns a list of sub-elements of specified name  node.getChildNodes()                         //returns a list of all child nodes |

**2. Read XML with DOM parser**

In below example code, I am assuming that user is already aware of the structure of employees.xml file (it’s nodes and attributes); So example directly start fetching information and start printing it in console. In real life application, we will use this information for some real purpose rather than printing it on console and leave.

|  |
| --- |
| //Get Document Builder  DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();  DocumentBuilder builder = factory.newDocumentBuilder();    //Build Document  Document document = builder.parse(new File("employees.xml"));    //Normalize the XML Structure; It's just too important !!  document.getDocumentElement().normalize();    //Here comes the root node  Element root = document.getDocumentElement();  System.out.println(root.getNodeName());    //Get all employees  NodeList nList = document.getElementsByTagName("employee");  System.out.println("============================");    for (int temp = 0; temp < nList.getLength(); temp++)  {   Node node = nList.item(temp);   System.out.println("");    //Just a separator   if (node.getNodeType() == Node.ELEMENT\_NODE)   {      //Print each employee's detail      Element eElement = (Element) node;      System.out.println("Employee 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("Location : "    + eElement.getElementsByTagName("location").item(0).getTextContent());   }  } |

Program Output:

|  |
| --- |
| employees  ============================    Employee id : 111  First Name : Lokesh  Last Name : Gupta  Location : India    Employee id : 222  First Name : Alex  Last Name : Gussin  Location : Russia    Employee id : 333  First Name : David  Last Name : Feezor  Location : USA |

**3. Read data to POJO objects**

Another real life application’s requirement might be populating the DTO objects with information fetched in above example code. I wrote a simple program to help you understand how it can be done easily.

Let’s say we have to populate Employee objects which is defined as below.

|  |
| --- |
| public class Employee  {     private Integer id;     private String firstName;     private String lastName;     private String location;       //Setters and Getters       @Override     public String toString()     {        return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", location=" + location + "]";     }  } |

Now look at the example code to populate employee objects list. Its just as simple as inserting few lines in between the code, and then copy the values in DTOs instead of console.

Java program to read XML file with DOM parser.

|  |
| --- |
| public class PopulateDTOExamplesWithParsedXML  {     public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException     {          List<Employee> employees = parseEmployeesXML();          System.out.println(employees);     }       private static List<Employee> parseEmployeesXML() throws ParserConfigurationException, SAXException, IOException     {        //Initialize a list of employees        List<Employee> employees = new ArrayList<Employee>();        Employee employee = null;          DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();        DocumentBuilder builder = factory.newDocumentBuilder();        Document document = builder.parse(new File("employees.xml"));        document.getDocumentElement().normalize();        NodeList nList = document.getElementsByTagName("employee");        for (int temp = 0; temp < nList.getLength(); temp++)        {           Node node = nList.item(temp);           if (node.getNodeType() == Node.ELEMENT\_NODE)           {              Element eElement = (Element) node;              //Create new Employee Object              employee = new Employee();              employee.setId(Integer.parseInt(eElement.getAttribute("id")));              employee.setFirstName(eElement.getElementsByTagName("firstName").item(0).getTextContent());              employee.setLastName(eElement.getElementsByTagName("lastName").item(0).getTextContent());              employee.setLocation(eElement.getElementsByTagName("location").item(0).getTextContent());                //Add Employee to list              employees.add(employee);           }        }        return employees;     }  } |

Program Output.

|  |
| --- |
| [Employee [id=111, firstName=Lokesh, lastName=Gupta, location=India],  Employee [id=222, firstName=Alex, lastName=Gussin, location=Russia],  Employee [id=333, firstName=David, lastName=Feezor, location=USA]] |

**4. Parse “unknown” xml with DOM parser**

Previous example shows the way we can iterate over an XML document parsed with known or little know structure to you, while you are writing the code. In some cases, we may have to write the code in such a way such that even if there is some differences in assumed XML structure while coding, program must work without failure.

Here we are iterating over all elements present in XML document tree. we can add our knowledge and modify the code such that as soon as we get required information while traversing the tree, we just use it.

|  |
| --- |
| public class ParseUnknownXMLStructure  {     public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException     {        //Get Document Builder        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();        DocumentBuilder builder = factory.newDocumentBuilder();          //Build Document        Document document = builder.parse(new File("employees.xml"));          //Normalize the XML Structure; It's just too important !!        document.getDocumentElement().normalize();          //Here comes the root node        Element root = document.getDocumentElement();        System.out.println(root.getNodeName());          //Get all employees        NodeList nList = document.getElementsByTagName("employee");        System.out.println("============================");          visitChildNodes(nList);     }       //This function is called recursively     private static void visitChildNodes(NodeList nList)     {        for (int temp = 0; temp < nList.getLength(); temp++)        {           Node node = nList.item(temp);           if (node.getNodeType() == Node.ELEMENT\_NODE)           {              System.out.println("Node Name = " + node.getNodeName() + "; Value = " + node.getTextContent());              //Check all attributes              if (node.hasAttributes()) {                 // get attributes names and values                 NamedNodeMap nodeMap = node.getAttributes();                 for (int i = 0; i < nodeMap.getLength(); i++)                 {                     Node tempNode = nodeMap.item(i);                     System.out.println("Attr name : " + tempNode.getNodeName()+ "; Value = " + tempNode.getNodeValue());                 }                 if (node.hasChildNodes()) {                    //We got more childs; Let's visit them as well                    visitChildNodes(node.getChildNodes());                 }             }           }        }     }  } |

Program Output.

|  |
| --- |
| employees  ============================  Node Name = employee; Value =          Lokesh          Gupta          India    Attr name : id; Value = 111  Node Name = firstName; Value = Lokesh  Node Name = lastName; Value = Gupta  Node Name = location; Value = India  Node Name = employee; Value =          Alex          Gussin          Russia    Attr name : id; Value = 222  Node Name = firstName; Value = Alex  Node Name = lastName; Value = Gussin  Node Name = location; Value = Russia  Node Name = employee; Value =          David          Feezor          USA    Attr name : id; Value = 333  Node Name = firstName; Value = David  Node Name = lastName; Value = Feezor  Node Name = location; Value = USA |