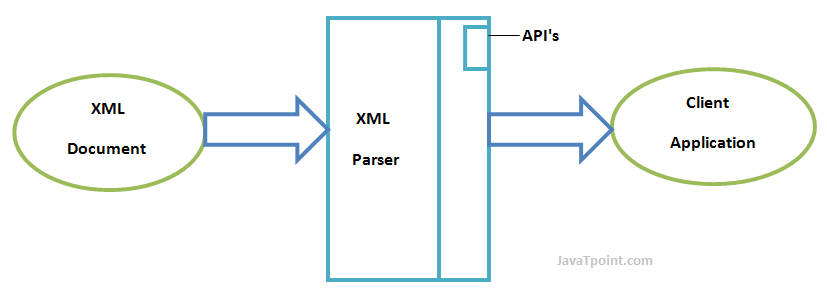
Top of Form

#### XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



#### Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

#### DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

#### Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

#### Advantages

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

#### Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

#### SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

#### Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

#### Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

#### Disadvantages

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

#### XML DOM

#### What is XML DOM

DOM is an acronym stands for Document Object Model. It defines a standard way to access and manipulate documents. The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications. The Document Object Model can be used with any programming language.

XML DOM defines a standard way to access and manipulate XML documents.

#### What does XML DOM

The XML DOM makes a tree-structure view for an XML document.

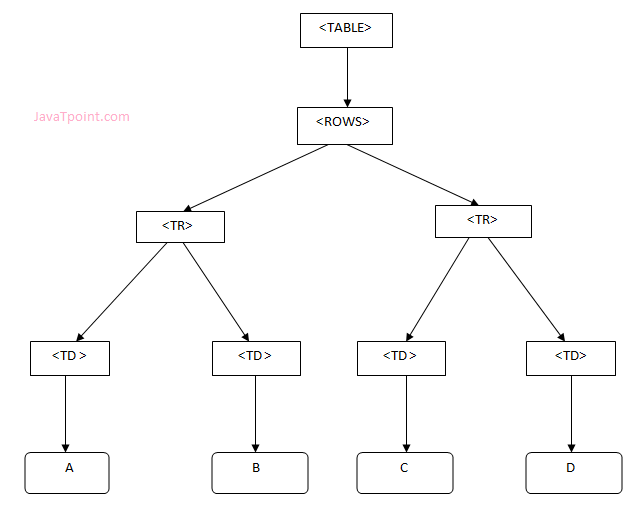
We can access all elements through the DOM tree.

We can modify or delete their content and also create new elements. The elements, their content (text and attributes) are all known as nodes.

For example, consider this table, taken from an HTML document:

1. **<TABLE>**
2. **<ROWS>**
3. **<TR>**
4. **<TD>**A**</TD>**
5. **<TD>**B**</TD>**
6. **</TR>**
7. **<TR>**
8. **<TD>**C**</TD>**
9. **<TD>**D**</TD>**
10. **</TR>**
11. **</ROWS>**
12. **</TABLE>**

The Document Object Model represents this table like this:



#### XML DOM Example : Load XML File

Let's take an example to show how an XML document ("note.xml") is parsed into an XML DOM object.

This example parses an XML document (note.xml) into an XML DOM object and extracts information from it with JavaScript.

Let's see the XML file that contains message.

note.xml

1. **<?xml** version="1.0" encoding="ISO-8859-1"**?>**
2. **<note>**
3. **<to>**sonoojaiswal@javatpoint.com**</to>**
4. **<from>**vimal@javatpoint.com**</from>**
5. **<body>**Hello XML DOM**</body>**
6. **</note>**

Let's see the HTML file that extracts the data of XML document using DOM.

xmldom.html

1. <!DOCTYPE html**>**
2. **<html>**
3. **<body>**
4. **<h1>**Important Note**</h1>**
5. **<div>**
6. **<b>**To:**</b>** **<span** id="to"**></span><br>**
7. **<b>**From:**</b>** **<span** id="from"**></span><br>**
8. **<b>**Message:**</b>** **<span** id="message"**></span>**
9. **</div>**
10. **<script>**
11. if (window.XMLHttpRequest)
12. {// code for IE7+, Firefox, Chrome, Opera, Safari
13. xmlhttp=new XMLHttpRequest();
14. }
15. else
16. {// code for IE6, IE5
17. xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
18. }
19. xmlhttp.open("GET","note.xml",false);
20. xmlhttp.send();
21. xmlDoc=xmlhttp.responseXML;
22. document.getElementById("to").innerHTML=
23. xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
24. document.getElementById("from").innerHTML=
25. xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
26. document.getElementById("message").innerHTML=
27. xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
28. **</script>**
29. **</body>**
30. **</html>**

**Output:**

#### Important Note

**To:** sonoojaiswal@javatpoint.com

**From:** vimal@javatpoint.com

**Message:** Hello XML DOM

Bottom of Form

Top of Form

#### REST

**REST** is the acronym for **REpresentational State Transfer**. REST is an **architectural style** for developing applications that can be accessed over the network.

#### Restful Web Services

Restful Web Services is a **stateless client-server** architecture where web services are resources and can be identified by their URIs.

REST Client applications can use HTTP GET/POST methods to invoke Restful web services. REST doesn’t specify any specific protocol to use, but in almost all cases it’s used over HTTP/HTTPS. When compared to SOAP web services, these are lightweight and doesn’t follow any standard. We can use XML, JSON, text or any other type of data for request and response.

#### Java RESTful Web Services API

Java API for RESTful Web Services (JAX-RS) is the Java API for creating REST web services. JAX-RS uses annotations to simplify the development and deployment of web services. JAX-RS is part of JDK, so you don’t need to include anything to use it’s annotations.

#### Restful Web Services Annotations

Some of the important JAX-RS annotations are:

* @Path: used to specify the relative path of class and methods. We can get the URI of a webservice by scanning the Path annotation value.
* @GET, @PUT, @POST, @DELETE and @HEAD: used to specify the HTTP request type for a method.
* @Produces, @Consumes: used to specify the request and response types.
* @PathParam: used to bind the method parameter to path value by parsing it.

#### Restful Web Services and SOAP

1. SOAP is a protocol whereas REST is an architectural style.
2. SOAP server and client applications are tightly coupled and bind with the WSDL contract whereas there is no contract in REST web services and client.
3. Learning curve is easy for REST when compared to SOAP web services.
4. REST web services request and response types can be XML, JSON, text etc. whereas SOAP works with XML only.
5. JAX-RS is the Java API for REST web services whereas JAX-WS is the Java API for SOAP web services.

#### REST API Implementations

There are two major implementations of JAX-RS API.

1. **Jersey**: [Jersey](https://jersey.github.io/) is the reference implementation provided by Sun. For using Jersey as our JAX-RS implementation, all we need to configure its servlet in web.xml and add required dependencies. Note that JAX-RS API is part of JDK not Jersey, so we have to add its dependency jars in our application.
2. **RESTEasy**: [RESTEasy](https://resteasy.github.io/) is the JBoss project that provides JAX-RS implementation.

Bottom of Form

Top of Form

**Video Lectures :**

**Reference Links:**

<https://www.tutorialspoint.com/xml/xml_parsers.htm>

<https://www.tutorialspoint.com/java_xml/java_xml_parsers.htm>

<https://www.w3schools.com/php/php_xml_parsers.asp>

<https://www.w3schools.com/xml/xml_parser.asp>

<http://www.stylusstudio.com/xml/parser.html>

Bottom of Form