# Web Technologies Extra Assignment

**By Alle Pranav Sudhan**
**21WU0102058**
**AI & DS- B2**

## 1Q. Explain XML parsers in detail.

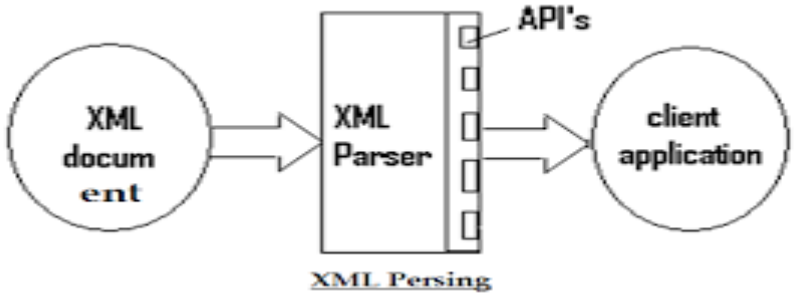
### XML:

XML full form is eXtensible Markup language and it was designed to store and transport data. It is designed so that both humans and machines can read.
XML has an important role in many different IT systems, it is used for distributing data over the Internet and it is a markup language like HTML.

### XML Parsers:

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 checks that the document is well formatted.

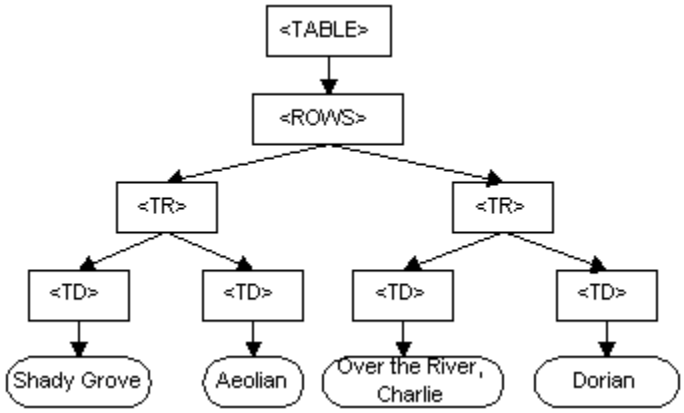



Types of XML Parsers:

There are two types of XML parsers and they are

1. DOM (Document Object Model)
2. SAX ( Simple API for XML )

- **1. DOM (Document Object Model)**
  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.
- A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information about the original XML document by invoking methods on this document object
- A standard object model and interface for XML.
- These are some typical DOM properties:
  1. x.nodeName - the name of x
  2. x.nodeValue - the value of x
  3. x.parentNode - the parent node of x
  4. x.childNodes - the child nodes of x
  5. x.attributes - the attributes nodes of x
- DOM Parser has a tree-based structure as shown below




| Advantages | Disadvantages |
|---|---|
| It supports both read and write operations and the API is very simple to use. | It is memory inefficient. (consumes more memory because the whole XML document needs to be loaded into memory). |
| . It is preferred when random access to widely separated parts of a document is required. | It is comparatively slower than other parsers. |

- Example:

```
<html>
<body>

<p id="demo"></p>

<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>Everyday Italian</title>" +
"<author>Giada De Laurentiis</author>" +
"<year>2005</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>

</body>
</html>
```

**2. SAX ( Simple API for XML )**

SAX parsers work on each element of the XML document sequentially, issuing parsing events while passing through the input stream in a single pass.
It is a programming interface for processing XML files based on events and has a very different way of reading XML code.
It does not create any internal structure. 2.

Clients do not know what methods to call, they just override 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
SAX parser does nothing with the data. It is up to the SAX parser's user to decide. The SAX events include (among others) as follows:

1. XML Text Nodes
2. XML Element Starts and Ends
3. XML Processing Instructions
4. XML Comment

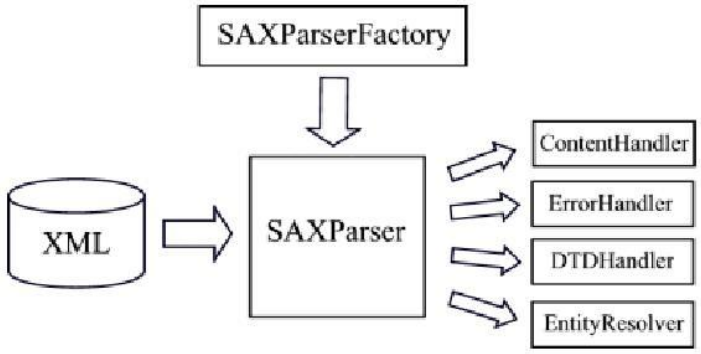



**Figure 2: SAX 2.0 Parsing**

| Advantages | Disadvantages |
|---|---|
| It is simple and memory efficient. | It is event-based so its API is less intuitive |
| It is very fast and works for huge documents. | Clients never know the full information because the data is broken into pieces |

**Example:**

```
public class SAXParserTest {
    public static void main(String[] args){
          try {
              //File Path
          String filePath = "D: classInfo.xml";

              //Create file object.
              File inputFile = new File(filePath);

              //Get SAXParserFactory instance.
              SAXParserFactory factory=SAXParserFactory.newInstance()

              //Get SAXParser object from SAXParserFactory instance.
              SAXParser saxParser = factory.newSAXParser();

              //Create StudentHandler object.
              StudentHandler studentHandler = new StudentHandler();

              //Parse the XML file.
              saxParser.parse(inputFile, studentHandler);
          } catch (Exception e) {
             e.printStackTrace();
          }
       }
}
```

**2Q. Discuss database connectivity, sessions and cookies both in PHP and JSP.**

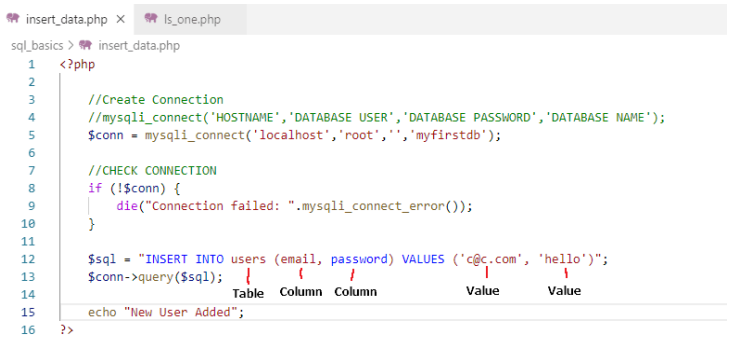
**PHP:**

PHP full form is Hypertext Preprocessor and it is widely used because of its open source and free to download scripting language and the scripts are executed on the server.

**Database connectivity:**

PHP offers various methods to connect to databases, such as MySQL, PostgreSQL, SQLite, etc. The most common method is using the MySQLi (MySQL Improved) or PDO (PHP Data Objects) extension.

It provides both an object-oriented and procedural interface for interacting with MySQL databases. Here's a basic example of connecting to a MySQL database




**Sessions in PHP:**

Sessions are a way to preserve data across subsequent HTTP requests. In PHP, sessions are often used to store user information and maintain their state throughout a browsing session.

PHP sessions rely on cookies or URLs to maintain session IDs.

1. **Creating a Session**

**In order to create a session, you must first call the PHP session_start function and then store your values in the $_SESSION array variable.**

```
<?php

session_start(); //start the PHP_session function

if(isset($_SESSION['page_count']))
{
     $_SESSION['page_count'] += 1;
}
else
{
     $_SESSION['page_count'] = 1;
}
 echo 'You are visitor number ' . $_SESSION['page_count'];

?>
```

2. **Destroying Session Variables**

The session_destroy() function is used to destroy the whole Php session variables.

```
<?php

 session_destroy(); //destroy entire session

?>
```

**Cookies in PHP**

Cookies are small pieces of data that are stored on the client-side (user's browser). PHP allows you to set and retrieve cookies, which can be useful for storing user preferences or session information.

Personalizing the user experience – this is achieved by allowing users to select their preferences.The page requested that follow are personalized based on the set preferences in the cookies

**1. Creating Cookies**

```
<?php

setcookie(cookie_name, cookie_value, [expiry_time], [cookie_path],
[domain], [secure], [httponly]);

?>
```

**JSP:**

JSP full form is Java Server Pages, it is used for creating web applications and to create dynamic web content. It is used to insert JAVA code to HTML pages and is an advanced version of Servlet Technology.

**Database connectivity:**

JSP can connect to databases using Java Database Connectivity (JDBC). JDBC provides a standard API for Java applications to interact with various databases.

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Guru Database JSP1</title>
</head>
<body>

 <sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost/GuruTest"
     user="gururoot"  password="guru"/>

<sql:query dataSource="${snapshot}" var="result">
SELECT * from guru_test;
</sql:query>
```

**Session in JSP:**

JSP makes use of the servlet provided HttpSession Interface. This interface provides a way to identify a user across.

By default, JSPs have session tracking enabled and a new HttpSession object is instantiated for each new client automatically. Disabling session tracking requires explicitly turning it off by setting the page directive session attribute to false as follows

```
<%
  // Get session creation time.
  Date createTime = new Date(session.getCreationTime());
  // Get last access time of this Webpage.
  Date lastAccessTime = new Date(session.getLastAccessedTime());
  String title = "Welcome Back to my website";
  Integer visitCount = new Integer(0);
  String visitCountKey = new String("visitCount");
  String userIDKey = new String("userID");
  String userID = new String("ABCD");
  // Check if this is new comer on your Webpage.
  if (session.isNew() ){
    title = "Welcome to my website";
    session.setAttribute(userIDKey, userID);
    session.setAttribute(visitCountKey,  visitCount);
  }
```

**Cookies in JSP:**

Cookies are text files stored on the client computer and they are kept for various information tracking purposes. JSP transparently supports HTTP cookies using underlying servlet technology.

**1. Reading a Cookie**

```
cookies = request.getCookies();

for (int i = 0; i < cookies.length; i++) {
        cookie = cookies[i];
        out.print("Name : " + cookie.getName( ) + ",  ");
        out.print("Value: " + cookie.getValue( )+" <br/>");
      }
```

**1. Deleting a Cookie**

```
cookies = request.getCookies();
cookie.setMaxAge(0);
 response.addCookie(cookie);
```

**3Q. Explain different jQuery selectors with a total of 10 different examples.**

**JQuery:**

JQuery is a small, lightweight and fast JavaScript library, it is platform-independent. It literally means “write less do more”, it simplifies AJAX calls and DOM manipulations.

**JQuery Selectors:**

JQuery Selectors are used to select and manipulate HTML elements.

With jQuery selectors, you can find or select HTML elements based on their id, classes, attributes, types and much more from a DOM.

All jQuery selectors start with a dollar sign and parenthesis e.g. $(). It is known as the factory function

**Different JQuery Selectors are:**

1. *** Selector:** It is used to select all elements.
2. **#id Selector: It will select the element with id="firstname"**
3. **.class Selector: It will select all elements with class="primary"**
4. **Class,.class Selector: It will select all elements with the class "primary" or "secondary"**
5. **:first Selector: This will select the first p element**
6. **:last-of-type Selector: It will select all p elements that are the last p element of their parent**
7. **:nth-lastchild(n) Selector: This will select all p elements that are the 2nd child of their parent, counting from the last child**
8. **parent > child Selector: It will select all p elements that are a direct child of a div element**
9. **parent descendant Selector: It will select all p elements that are descendants of a div element**
10. **:image Selector: It will select all input elements with type="image"**