|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Session** | **2024-25 (ODD)** | | **Course Name** | **Web Technology Lab** | |
| **Semester** | **3** | | **Course Code** | **23CT1301** | |
| **Roll No** | **53** | | **Name of Student** | **Kartik Sheshraoji Tale** | |
|  |  | |  |  |  |
| Practical Number | | 5 | | | |
| Course Outcome | | 1. Understand various internet technologies. 2. Design the web pages using HTML and CSS. 3. Implement the XML technology to store the data. 4. Develop the interactive web pages using JavaScript. | | | |
| Aim | | Aim: [A] Introduction to XML. Program to demonstrate the use of External and Internal DTD. (Write an XML file which will display the employee information which includes the following: 1) Employee ID 2) Name of Employee 3) Department of Employee 4) Designation of Employee 5) Email of Employee 6) Salary of Employee  [B] Create XML for employee information and access second student's data using DOM. | | | |
| Problem Definition | | The problem is to demonstrate how XML can be used to store and validate structured employee information using internal and external DTDs, and to implement DOM parsing in JavaScript to access and display the second employee’s details from an XML file. | | | |
| Theory  (100 words) | | XML (Extensible Markup Language) is a widely used standard for storing and transporting structured information in a platform-independent manner. It represents data hierarchically, making it both human-readable and machine-processable. To ensure correctness, a **DTD (Document Type Definition)** defines the structure, allowed elements, and attributes in an XML file. Internal DTDs are declared inside the XML, whereas external DTDs are maintained separately and linked. To retrieve and manipulate XML data, the **DOM (Document Object Model)** provides a tree-based structure where nodes can be accessed using programming languages such as JavaScript. DOM parsing enables dynamic extraction of specific data, like employee details. | | | |
| Procedure and Execution  (100 Words) | | 1. Step for Implementation:    1. Create an employee XML file with all required fields: ID, Name, Department, Designation, Email, Salary.    2. Add an internal DTD inside the XML for validation.    3. Create an external DTD file and link it with the XML.    4. Prepare another XML file containing multiple employee records.    5. Write a JavaScript DOM program to parse this XML and access the second employee’s data.    6. Display the extracted data on a webpage using HTML and CSS formatting. | | | |
| Code:  <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Employee Information</title>    <style>      body {        font-family: Arial, sans-serif;        background: white;        color: black;        margin: 20px;      }      h1 {        text-align: center;        color: blue;      }      button {        display: block;        margin: 15px auto;        padding: 10px 20px;        font-size: 16px;        background: blueviolet;        color: whitesmoke;        border: none;        border-radius: 6px;        cursor: pointer;        transition: 0.3s;      }      button:hover {        background: honeydew;      }      h2 {        color: navajowhite;        margin-top: 30px;        text-align: center;      }      .employee-card {        max-width: 400px;        margin: 20px auto;        background: white;        padding: 20px;        border-radius: 8px;        box-shadow: 0px 4px 6px rgba(0,0,0,0.1);      }      .employee-card p {        font-size: 16px;        margin: 8px 0;      }      .employee-card strong {        color: blanchedalmond;      }    </style>  </head>  <body>    <h1>Employee Information</h1>    <button onclick="getSecondEmployee()">Get Second Employee's Info</button>    <h2>Second Employee's Data:</h2>    <div id="second-employee-info" class="employee-card"></div>    <!-- XML data embedded inside HTML -->    <script id="xmlData" type="text/xml">      <employees>        <employee>          <empID>101</empID>          <name>Rahul Sharma</name>          <department>IT</department>          <designation>Software Engineer</designation>          <email>rahul.sharma@example.com</email>          <salary>55000</salary>        </employee>        <employee>          <empID>102</empID>          <name>Priya Verma</name>          <department>HR</department>          <designation>HR Manager</designation>          <email>priya.verma@example.com</email>          <salary>60000</salary>        </employee>        <employee>          <empID>103</empID>          <name>Amit Patel</name>          <department>Finance</department>          <designation>Accountant</designation>          <email>amit.patel@example.com</email>          <salary>50000</salary>        </employee>      </employees>    </script>    <script>      function loadXML() {        const xmlText = document.getElementById("xmlData").textContent;        const parser = new DOMParser();        return parser.parseFromString(xmlText, "application/xml");      }      function getSecondEmployee() {        const xmlDoc = loadXML();        const employees = xmlDoc.getElementsByTagName("employee");        if (employees.length >= 2) {          const secondEmployee = employees[1];          const id = secondEmployee.getElementsByTagName("empID")[0].textContent;          const name = secondEmployee.getElementsByTagName("name")[0].textContent;          const department = secondEmployee.getElementsByTagName("department")[0].textContent;          const designation = secondEmployee.getElementsByTagName("designation")[0].textContent;          const email = secondEmployee.getElementsByTagName("email")[0].textContent;          const salary = secondEmployee.getElementsByTagName("salary")[0].textContent;          document.getElementById("second-employee-info").innerHTML = `            <p><strong>ID:</strong> ${id}</p>            <p><strong>Name:</strong> ${name}</p>            <p><strong>Department:</strong> ${department}</p>            <p><strong>Designation:</strong> ${designation}</p>            <p><strong>Email:</strong> ${email}</p>            <p><strong>Salary:</strong> ${salary}</p>          `;        } else {          alert("There are less than 2 employees in the data.");        }      }    </script>  </body>  </html> | | | |
| Output: | | | |
| Output Analysis | | Employee XML was successfully validated using both internal and external DTDs.  DOM parsing correctly extracted the second employee’s details such as ID, Name, Department, Designation, Email, and Salary.  Data was displayed in a structured, readable format in the browser. | | | |
| Link of student Github profile where lab assignment has been uploaded | | <https://github.com/kartiktale12/WebTechnology> | | | |
| Conclusion | | The experiment shows that XML with DTDs ensures structured data validation, and DOM parsing allows easy retrieval of specific employee details dynamically | | | |
| Plag Report (Similarity index < 12%) | | 9% | | | |
| Date | | 25/08/2025 | | | |