Name : ShreeGanesh Purohit
RegNo : 2447152
Subject : MCA WEBSTACK LAB

Experiment 5

XML Transformation and Validation Process

Domain : Personal Stock Portfolio Planner

1. Overview

This report documents the transformation and validation process of an XML document representing Nifty 50 stock data. The process involved creating an XML document based on a predefined DTD structure, validating it against the schema, and testing various scenarios to identify and resolve potential issues.

2. XML Document Creation

The XML document was created to represent stock data for 15 Nifty 50 companies. Each company's data was encapsulated within a stock symbol element, which included the company name, sector, current price, P/B ratio, P/E ratio, ROE, market capitalization, and date added.

```
<!ELEMENT stocks (stocksymbole+)>
<!ELEMENT stocksymbole (companyname, sector, currentprice, pbratio, peratio, roe, marketcapitalisation, dateadded)>
<!ELEMENT companyname (#PCDATA)>
<!ELEMENT sector (#PCDATA)>
<!ELEMENT currentprice (#PCDATA)>
<!ELEMENT pbratio (#PCDATA)>
<!ELEMENT peratio (#PCDATA)>
<!ELEMENT roe (#PCDATA)>
<!ELEMENT marketcapitalisation (#PCDATA)>
<!ELEMENT dateadded (#PCDATA)>
```

3. XSD Schema Design

To ensure the data in the XML document adheres to certain constraints, an XSD (XML Schema Definition) was designed. The XSD enforces data types and structures beyond the basic validation provided by the DTD.

Purpose of XSD Schema:

- **Data Type Enforcement**: Ensures that elements like currentprice, pbratio, peratio, roe, and marketcapitalisation are of specific numeric types.

- **Structural Validation**: Ensures the presence and order of elements within each stocksymbole.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="stocks">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="stocksymbole" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="companyname" type="xs:string"/>
              <xs:element name="sector" type="xs:string"/>
              <xs:element name="currentprice" type="xs:float"/>
              <xs:element name="pbratio" type="xs:float"/>
              <xs:element name="peratio" type="xs:float"/>
              <xs:element name="roe" type="xs:float"/>
              <xs:element name="marketcapitalisation" type="xs:long"/>
              <xs:element name="dateadded" type="xs:date"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4. Transformation with XSLT

An XSLT (Extensible Stylesheet Language Transformations) stylesheet was used to transform the XML data into a more human-readable format, such as HTML or another XML structure.

Purpose of the XSL Stylesheet:

- **Transformation**: Converts the raw XML data into a different format for presentation or further processing.
- **Data Extraction**: Allows selective extraction and formatting of XML content.

Sample XSLT Code:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
    <body>
      <h2>Nifty 50 Stock Data</h2>
      <table border="1">
```

```
        <tr>
          <th>Company Name</th>
          <th>Sector</th>
          <th>Current Price</th>
          <th>P/B Ratio</th>
          <th>P/E Ratio</th>
          <th>ROE (%)</th>
          <th>Market Capitalisation</th>
          <th>Date Added</th>
        </tr>
        <xsl:for-each select="stocks/stocksymbole">
        <tr>
          <td><xsl:value-of select="companyname"/></td>
          <td><xsl:value-of select="sector"/></td>
          <td><xsl:value-of select="currentprice"/></td>
          <td><xsl:value-of select="pbratio"/></td>
          <td><xsl:value-of select="peratio"/></td>
          <td><xsl:value-of select="roe"/></td>
          <td><xsl:value-of select="marketcapitalisation"/></td>
          <td><xsl:value-of select="dateadded"/></td>
        </tr>
        </xsl:for-each>
      </table>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

5. Validation Process

The XML document was validated against the XSD schema to ensure compliance.

Validation Steps:

1. **Schema Validation**: Ensuring that the XML document adheres to the structure and data types defined in the XSD.
2. **Error Detection**: Identifying and resolving any issues where the XML content violated schema rules.
3. **Testing Scenarios**: Various scenarios were tested, including:
   - **Correct Data**: Ensuring valid XML files passed validation.
   - **Missing Elements**: Checking how the schema handles missing required elements.
   - **Invalid Data Types**: Testing XML files with incorrect data types, such as strings in numeric fields.

Errors Encountered:

- **Missing Elements**: If a required element, such as currentprice, was missing, the validation failed.
- **Data Type Mismatch**: When a string was used in place of a numeric type (e.g., currentprice as "abcd"), validation produced errors.
- **Date Format**: Dates that did not adhere to the YYYY-MM-DD format were flagged as errors.

6. Documentation

The process was documented meticulously, with attention given to explaining the purpose of each component (XSLT, XSD, and validation scripts). This documentation serves as a comprehensive guide for understanding the transformation and validation process.

Tools Used:

- **XML Editor**: For writing and editing XML documents.
- **xmllint**: Command-line tool for validating XML against XSD.
- **XSLT Processor**: For transforming XML into other formats.

Conclusion

This case study demonstrates the complete lifecycle of XML data transformation and validation, from creation to schema enforcement and error handling. The process ensures data integrity and showcases the application of XML-related technologies in a structured and practical manner.