Web Stack Development

Lab Exercise-5

Submission Date:

11th August 2024

Submitted to:

Dr. Cynthia T

Submitted by:

Eileen Maria Tom

(2447118)

Documentation of Lab Exercise:

Domain: Automobile Manufacturing-

1. carproduct.xml file

carproduct.xml > ...

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ProductDetails xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="carproduct.xsd">
                <?xml-stylesheet type="text/xsl" href="transform.xsl"?>

  <Product id="PDT001">
    <ProductID>CGS098</ProductID>
    <Name>Continental GTC Speed</Name>
    <Type>Sports Car</Type>
    <Price>2300000.00</Price>
    <ModelNo>2024</ModelNo>
    <Colour>Shiny Green</Colour>
    <Warranty>5</Warranty>
  </Product>
  <Product id="PDT002">
    <ProductID>BEA876</ProductID>
    <Name>Bentayga EWB Azure</Name>
    <Type>SUV</Type>
    <Price>2506000.00</Price>
    <ModelNo>2023</ModelNo>
    <Colour>Shiny Black</Colour>
    <Warranty>10</Warranty>
  </Product>
  <Product id="PDT003">
    <ProductID>SPX123</ProductID>
    <Name>Flying Spur X</Name>
    <Type>Car</Type>
    <Price>2000800.00</Price>
    <ModelNo>2019</ModelNo>
    <Colour>Navy Blue</Colour>
    <Warranty>3</Warranty>
  </Product>
</ProductDetails>
```

2. carproduct.xsd file

carproduct.xsd > xs:schema > xs:simpleType

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3
4     <xs:element name="ProductDetails">
5       <xs:complexType>
6         <xs:sequence>
7
8           <xs:element name="Product" maxOccurs="unbounded">
9             <xs:complexType>
10              <xs:sequence>
11                <xs:element name="ProductID" type="xs:string"/>
12                <xs:element name="Name" type="xs:string"/>
13                <xs:element name="Type" type="xs:string"/>
14                <xs:element name="Price" type="PriceType"/>
15                <xs:element name="ModelNo" type="ModelNoType"/>
16                <xs:element name="Colour" type="xs:string"/>
17                <xs:element name="Warranty" type="WarrantyType"/>
18              </xs:sequence>
19              <xs:attribute name="id" type="xs:string" use="required"/>
20            </xs:complexType>
21          </xs:element>
22        </xs:sequence>
23      </xs:complexType>
24    </xs:element>
25
26    <xs:simpleType name="PriceType">
27      <xs:restriction base="xs:decimal">
28        <xs:minInclusive value="0.01"/>
29      </xs:restriction>
30    </xs:simpleType>
```

```
31
32      <xs:simpleType name="ModelNoType">
33        <xs:restriction base="xs:integer">
34          <xs:minInclusive value="3"/>
35        </xs:restriction>
36      </xs:simpleType>
37
38      <xs:simpleType name="WarrantyType">
39        <xs:restriction base="xs:integer">
40          <xs:minInclusive value="0"/>
41          <xs:maxInclusive value="10"/>
42        </xs:restriction>
43      </xs:simpleType>
44    </xs:schema>
45
```

3. transform.xsl file

transform.xsl > xsl:stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <html>
            <head>
                <title>Product Details</title>
                <style>
                    table {
                        width: 100%;
                        border-collapse: collapse;
                    }
                    th, td {
                        border: 1px solid black;
                        padding: 8px;
                        text-align: left;
                    }
                    th {
                        background-color: white;
                    }
                </style>
            </head>
            <body>
                <h1>Product Details</h1>
                <table>
                    <tr>
                        <th>Product ID</th>
                        <th>Name</th>
                        <th>Type</th>
                        <th>Price</th>
                        <th>ModelNo</th>
                        <th>Colour</th>
                        <th>Warranty</th>
                    </tr>
                    <xsl:for-each select="ProductDetails/Product">
                        <tr>
                            <td><xsl:value-of select="ProductID"/></td>
                            <td><xsl:value-of select="Name"/></td>
                            <td><xsl:value-of select="Type"/></td>
                            <td><xsl:value-of select="Price"/></td>
                            <td><xsl:value-of select="ModelNo"/></td>
                            <td><xsl:value-of select="Colour"/></td>
                            <td><xsl:value-of select="Warranty"/></td>
                        </tr>
                    </xsl:for-each>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Summary of transformation and validation process for the XML, XSD and XSL files:

1. XML File (carproduct.xml)

It contains a root element <ProductDetails> which includes elements like <Product> elements and sub elements like <ProductID>, <Name>, <Type>, <Price>, <ModelNo>, <Colour> and <Warranty>. It uses carproduct.xsd and transform.xsl for validation and transformation respectively.

2. XSL Stylesheet (transform.xsl)

The XML style sheet helps to transform the XML data into an HTML table format and contains header and various HTML styles. The <xsl:for-each> tag is used to extract data and iterate over each <Product> element and generate rows according to the product details.

3. XML Schema (carproduct.xsd)

The XML schema definition provides the structure and data types for the XML file.

<ProductDetails>- It contains multiple <Product> elements like <ProductID>, <Name>, <Type>, <Price>, <ModelNo>, <Colour> and <Warranty>.

Potential Issues and Errors:

- XSD Validity

ModelNo- The restriction on ModelNoType is xs:integer with a minimum value of 3. However, a model numbers can include non-numeric characters like “2024" or "202A". Changing the type to `xs:string` might be more appropriate if model numbers are alphanumeric.

Price- The xs:decimal type with a minimum value of 0.01 should be fine for price values.

- Validation Against Schema

The XML file used follow the rules or constraints mentioned in the .xsd file. Any deviation or discrepancies in data types or element structures will result in validation errors.

- XSL Transformation

When the XML file is processed with the help of an XSL file, it should generate an html table with all the styling applied.

Solution:

Below is the table generated using the XSL file that transforms the XML data into a user-friendly HTML table format.

Product Details

Product ID	Name	Type	Price	ModelNo	Colour	Warranty
CGS098	Continental GTC Speed	Sports Car	2300000.00	2024	Shiny Green	5
BEA876	Bentayga EWB Azure	SUV	2506000.00	2023	Shiny Black	10
SPX123	Flying Spur X	Car	2000800.00	2019	Navy Blue	3

Purpose of XSL stylesheet, XSD schema and the script or program used for transformation and validation:

- XSL Stylesheet (transform.xsl)

The XSL stylesheet is used to transform the XML data into an HTML. It specifies the formatting and layout of how the data is to be presented. It iterates over each <Product> element in the XML and extracts the data.

- XSD Schema (carproduct.xsd)

The XSD schema is used to define the structure and constraints of the XML data and ensures that the XML document follows the specified rules. It also defines the element hierarchy, validates data types and constraints.

- Script used for Transformation and Validation

Validation Script helps to check if the XML document conforms to the constraints defined in the XSD file. Transformation Script converts the XML data into an HTML using the XSL stylesheet.

Self-Learning Component:

There is another way of validating an XML file without using an XSD file. This can be done using Python libraries like lxml and xmlschema.

```
[ ]: from lxml import etree

def validate_xml(xml_file, xsd_file):
    # Load and parse the XSD schema
    with open(xsd_file, 'r') as schema_file:
        schema_root = etree.parse(schema_file)
        schema = etree.XMLSchema(schema_root)

    # Create an XML parser with the schema
    parser = etree.XMLParser(schema=schema)

    # Parse the XML file
    try:
        etree.parse(xml_file, parser)
        print("XML is valid")
    except etree.XMLSyntaxError as e:
        print(f"XML is invalid: {e}")

xml_file = 'carproduct.xml'
xsd_file = 'carproduct.xsd'

validate_xml(xml_file, xsd_file)
```