Name : Ritam Maity

Register Number : 2447141

Class : MCA - A

Course : Web Stack Development

Date of submission : 11 August 2024

Lab Exercise 5

Summary report of transformation and validation process :

Issues encountered :

1. Error in the XSLT element

Reason – namespace spelling error in xsl:template:

payment.xsl

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3      xmlns:payment="http://www.example.com/payment">
4      <xsl:template match="/payment:transaction">
5          <html>
6              <head>
7                  <style>
```

Error:

This page contains the following errors:

error on line 5 at column 13: Start tag expected, '<' not found

Below is a rendering of the page up to the first error.

This document was created as the result of an XSL transformation. The line and column numbers given are from the transformed result.

Rectified:

payment.xsl

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3      xmlns:payment="http://www.example.com/payment">
4      <xsl:template match="/payment:transactions">
5          <html>
6              <head>
7                  <style>
```

Payment.xml

payment.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="payment.xsl" type="text/xsl" ?>
<payment:transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:payment="http://www.example.com/payment" xsi:schemaLocation="http://www.example.com/paym

    <payment:transaction id="t1">
        <payment:payer>
            <payment:name>Healthy Cleaning Service</payment:name>
            <payment:phone>156-456-7890</payment:phone>
            <payment:email>contact@healthycleaning.com</payment:email>
            <payment:address>123 Tiffin Street, Food City</payment:address>
        </payment:payer>
        <payment:details>
            <payment:amount>15.99</payment:amount>
            <payment:date>2024-08-10</payment:date>
            <payment:status>Completed</payment:status>
            <payment:method>Credit Card</payment:method>
            <payment:reference>HTS20240810XYZ</payment:reference>
        </payment:details>
        <payment:items>
            <payment:item>
                <payment:itemName>Cleaning Service Package</payment:itemName>
                <payment:itemPrice>10.00</payment:itemPrice>
            </payment:item>
            <payment:item>
                <payment:itemName>Additional Service</payment:itemName>
                <payment:itemPrice>5.99</payment:itemPrice>
            </payment:item>
        </payment:items>
    </payment:transaction>
```

payment.xml X payment.xsd payment.xsl

payment.xml

```xml
<payment:transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:payment="http://www.example.com/payment" xsi:schemaLocation="http://www.example.com/paym

    <payment:transaction id="t2">
        <payment:payer>
            <payment:name>Quick Grocery Delivery</payment:name>
            <payment:phone>987-654-3210</payment:phone>
            <payment:email>info@quickgrocerydeliv.com</payment:email>
            <payment:address>456 Snack Lane, Fastfood Town</payment:address>
        </payment:payer>
        <payment:details>
            <payment:amount>12.50</payment:amount>
            <payment:date>2024-08-11</payment:date>
            <payment:status>Pending</payment:status>
            <payment:method>PayPal</payment:method>
            <payment:reference>QBT20240811ABC</payment:reference>
        </payment:details>
        <payment:items>
            <payment:item>
                <payment:itemName>Grocery Package 1</payment:itemName>
                <payment:itemPrice>7.00</payment:itemPrice>
            </payment:item>
            <payment:item>
                <payment:itemName>Grocery Package 2</payment:itemName>
                <payment:itemPrice>5.50</payment:itemPrice>
            </payment:item>
        </payment:items>
    </payment:transaction>

    <payment:transaction id="t3">
        <payment:payer>
            <payment:name>Delightful Tiffin</payment:name>
            <payment:phone>234-567-8901</payment:phone>
            <payment:email>support@delightfultiffin.com</payment:email>
            <payment:address>789 Gourmet Avenue, Cuisine City</payment:address>
        </payment:payer>
```

Payment.xsd

payment.xsd

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" elementFormDefault="qualified" attributeFormDefault="unqualif
targetNamespace="http://www.example.com/payment" xmlns:payment="http://www.example.com/payment">

    <!-- The pattern for phone numbers -->
    <xs:simpleType name="phoneType">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{3}-\d{3}-\d{4}"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- The pattern for email addresses -->
    <xs:simpleType name="emailType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[a-zA-Z0-9._]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Limiting and restriction for amount -->
    <xs:simpleType name="amountType">
        <xs:restriction base="xs:decimal">
            <xs:minExclusive value="0"/>
            <xs:maxExclusive value="1000"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- the payer details complex type -->
    <xs:complexType name="payerType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="phone" type="payment:phoneType"/>
            <xs:element name="email" type="payment:emailType"/>
            <xs:element name="address" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
```

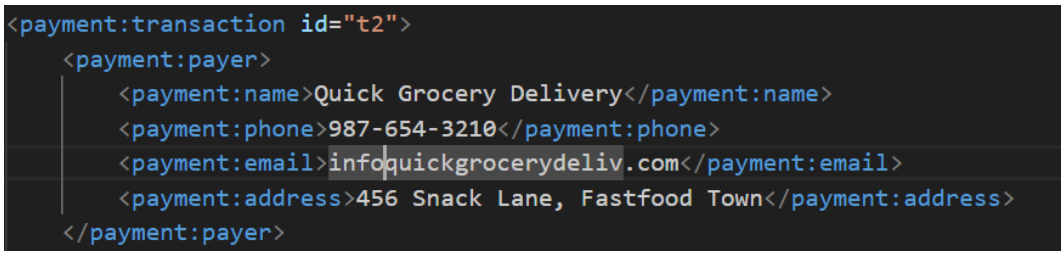
payment.xsd

```xml
targetNamespace="http://www.example.com/payment" xmlns:payment="http://www.example.com/payment">

    <!-- Define the transaction details complex type -->
    <xs:complexType name="transactionType">
        <xs:sequence>
            <xs:element name="payer" type="payment:payerType"/>
            <xs:element name="amount" type="payment:amountType"/>
            <xs:element name="date" type="xs:date"/>
            <xs:element name="status" type="xs:string"/>
            <xs:element name="method" type="xs:string"/>
            <xs:element name="reference" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required"/>
    </xs:complexType>

    <!-- Defining the root element -->
    <xs:complexType name="transactionsType">
        <xs:sequence>
            <xs:element name="transaction" type="payment:transactionType" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Declaration of the root element -->
    <xs:element name="transactions" type="payment:transactionsType"/>

</xs:schema>
```

Cases where xml violated XSD defined rules and shows errors:

1. When phone number has more than 10 numbers

2. When email does not follow the specified pattern



Documentation of the Solution

1. XML File (payment.xml)

The XML file defines a list of payment transactions. Each transaction includes details such as the payer's name, contact information, amount paid, payment date, status, method, and a reference number.

Key Elements:

- <transactions>: Root element containing all transaction records.
- <transaction>: Represents an individual transaction, identified by a unique id attribute.
- <payer>: Contains information about the person or entity making the payment.
- <amount>, <date>, <status>, <method>, <reference>: Various elements detailing the transaction's specifics.

2. XSL Stylesheet (payment.xsl)

The XSL file is used to transform the XML data into an HTML format, making it readable in a web browser. It processes each transaction and displays the information in a tabular format. The XSL file handles the layout, styling, and structure of the output HTML.

Purpose:

- **Transformation**: Convert XML data into a human-readable HTML format.
- **Styling**: Apply CSS styles to enhance readability, including table formatting, alternating row colors, and hover effects.

Key Features:

- **For-Each Loop**: Iterates through each transaction in the XML.
- **HTML Table**: Displays the transaction data in a structured format.
- **CSS Styles**: Ensures the HTML output is visually appealing and easy to read.

3. XSD Schema (payment.xsd)

The XSD file defines the structure and data types used in the XML file, ensuring that the XML adheres to the specified format.

Purpose:

- **Validation**: Ensures that the XML data is correctly structured and that data types are consistent.
- **Data Integrity**: Prevents incorrect data from being stored in the XML by defining constraints such as patterns for phone numbers and emails, and restricting values for elements like amount.

Key Features:

- **Simple Types**: Define custom patterns for phone numbers and email addresses.
- **Complex Types**: Define structures for transactions and payer information.
- **Constraints**: Ensure data validity, such as non-negative amounts and specific formats for dates and emails.

4. Transformation and Validation

- **Transformation**: The transformation is typically done using an XSLT processor, which applies the XSL stylesheet to the XML data, generating an HTML file.
- **Validation**: The XML file is validated against the XSD schema using an XML validator, ensuring the data structure complies with the defined schema.