GARAGE BOOKING WEBSITE VALIDATION

In this case study, we are working to turn data about garages into a format that is easy for users to read on a website. We have garage information in XML format, which is a type of data file. We need to convert this XML data into a neat HTML table using XSL (Extensible Stylesheet Language). This table will display details like the garage name, location, price, and availability.

We also need to make sure the XML data follows specific rules set by an XML Schema Definition (XSD). The XSD defines what the XML data should look like and what values are allowed. This helps ensure that the data is correct and follows the required format.

XSL Stylesheet (transform.xsl):

- **Purpose:** To transform XML data into a user-friendly HTML format. The XSL stylesheet formats the XML data into a well-structured HTML table and includes a header and footer for better visual presentation. It also provides a "Book" button for each garage entry.
- **Key Features:**
    - Converts XML data into an HTML table.
    - Styles the table for readability.
    - Includes a navigation button ("Book") linking to a booking page for each garage.
- **Key Sections:**
    - <xsl:stylesheet>: Defines the XSLT version and namespaces.
    - <xsl:template match="/">: Root template that creates HTML structure.
    - <xsl:for-each select="gb:garages/gb:garage">: Iterates over garage entries to populate the table.

XML Schema Definition (garage.xsd):

- **Purpose:** To define the structure, rules, and constraints for the XML data. It ensures that the XML data adheres to specific formats and values.
- **Key Features:**
    - Specifies required elements and their types.
    - Ensures data validity and consistency.
    - Optionally enforces unique constraints on certain data elements (though omitted in this version).
- **Key Sections:**
    - <xs:schema>: Defines the XSD namespace and schema.

- <xs:element name="garages">: Root element definition.
- <xs:complexType>: Defines the structure of elements within the garages element.

XML Data File (garage.xml):

- **Purpose:** To provide the data that will be transformed into HTML by the XSL stylesheet.
- **Key Features:**
  - Contains information about garages, including name, location, price, and availability.
- **Key Sections:**
  - <garages>: Root element.
  - <garage>: Contains individual garage details

Transformation and Validation Process

Transformation:

- **Tool Used:** XSLT Processor (such as Saxon or XSLT transformations in web browsers).
- **Process:**
  1. Load XML data (garage.xml).
  2. Apply the XSL stylesheet (transform.xsl) to convert XML data into HTML format.
  3. The transformation includes creating a structured HTML table from the XML data, applying styles for readability, and adding navigation functionality.

Validation:

- **Tool Used:** XML Validator (such as online validators or integrated development environments like VS Code with XSD support).
- **Process:**
  1. Validate XML data against the XSD schema (garage.xsd) to ensure it conforms to defined structure and rules.
  2. Check for errors related to structure, content, and constraints.

Errors and Issues Encountered:

1. **XSD Validation Issues:**
    - **Errors:**
        - **Element 'unique' Invalid:** The unique constraint in XSD was misplaced. Fixed by removing it as per current requirements.
        - **Entity Not Declared:** Entity references such as copy were included incorrectly. Fixed by removing or correcting the entity references.
        - **Invalid Content in 'schema':** Corrected by ensuring all elements are in proper sequence and valid.
2. **XSL Transformation Issues:**
    - **Errors:**
        - **Malformed href Attributes:** Addressed by ensuring href values in anchor tags are properly encoded and do not contain invalid characters.
        - **Entity Reference Issues:** Fixed by removing or declaring required entities correctly.

Testing Scenarios:

1. **Valid Data Test:**
    - Used valid XML data that matches the schema. Transformation produced correct HTML output with the garage data displayed properly in a table format.
2. **Invalid Data Tests:**
    - **Missing Required Elements:** XML without required elements (like name) was used. Errors in XML validation were correctly flagged.
    - **Invalid Data Types:** XML data with incorrect data types (e.g., non-numeric price values) was tested. The XSD validation identified these issues.
    - **Invalid Constraints:** Testing data with duplicate name values was performed. If unique constraints were used, it would have flagged duplicate entries.

Scripts/Programs Used:

- **XSLT Processor:** Used to apply the XSL stylesheet to the XML data.
- **XML Validator:** Tools or IDEs used for validating XML data against the XSD schema.

Conclusion

The provided solution includes a detailed approach to transforming XML data into HTML and validating it using XSD. All errors encountered were addressed, and the final output adheres to the defined schema and user requirements. The transformation and validation process was tested thoroughly to ensure robustness and reliability.