UNIVERSITY OF SRI JAYEWARDENEPURA

Faculty of Technology

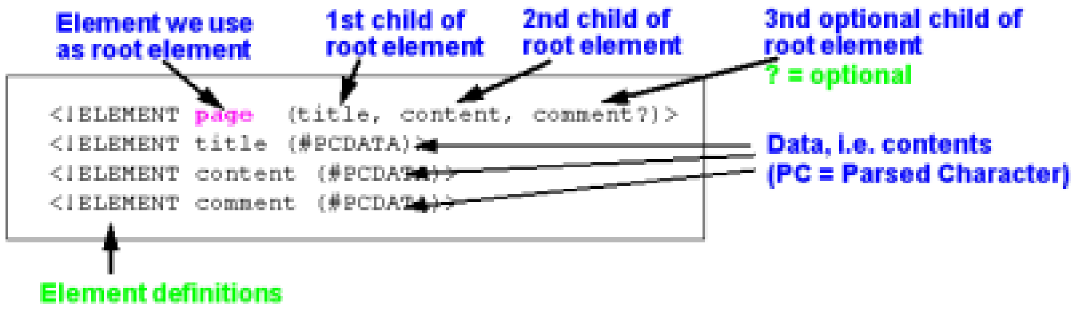
Department of Information and Communication Technology

ITC 3013 - Service Oriented Architecture & Web Services

Lab Sheet 2 – DTD & XML Schema

What is DTD?

XML (eXtensible Markup Language) Document Type Definition (DTD) is a way to describe the structure and rules of an XML document. It serves as a blueprint or schema that defines the elements, attributes, and relationships within an XML document. DTDs are used to ensure that XML documents conform to a predefined structure, making it easier to validate, interpret, and exchange data between different systems.


Element we use as root element
1st child of root element
2nd child of root element
3nd optional child of root element
? = optional
Data, i.e. contents
(PC = Parsed Character)
Element definitions

```
<!ELEMENT page   (title, content, comment?)>
<!ELEMENT title  (#PCDATA)>
<!ELEMENT content  (#PCDATA)>
<!ELEMENT comment  (#PCDATA)>
```

Here are some key points about XML DTDs:

1. Element Definitions:
   DTD defines the elements that can appear in an XML document, including their order and nesting relationships.

2. Attribute Definitions:
   DTD specifies the attributes that each element can have and defines their data types or values.

3. Entity Declarations:
   DTD allows you to define entities, which can be used to represent reusable pieces of text or special characters. This promotes code reusability and helps avoid redundancy.

4. Element Content Models:
   DTD defines the allowable structure of an element's content, whether it's empty, contains only text, contains other elements, or follows a specific pattern.

5. Validation:
   XML documents can be validated against their associated DTD to ensure that they conform to the defined structure and rules. This validation process helps catch errors and inconsistencies in data.

6. External and Internal DTDs:
   DTDs can be either internal (defined within the XML document) or external (referenced from an external file). External DTDs promote reusability and separation of concerns.

A simple example of an XML document with an associated DTD:

XML Document:

```
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

DTD (note.dtd):

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

In this example, the DTD defines the structure of a simple note element containing sub-elements like 'to', 'from', 'heading', and 'body'. The content model (#PCDATA) indicates that the element's content can only contain parsed character data (text).

While XML DTDs are useful for basic validation, more complex XML structures and validations are often achieved using alternatives like XML Schema or Relax NG, which offer more advanced features and extensibility.

What are the benefits of using DTD in XML documents?

1. Validation:
   DTDs ensure that XML documents conform to a predefined structure, helping to catch errors and maintain data integrity.

2. Interoperability:
   DTDs establish a common structure for XML data, promoting seamless data exchange between different systems.

3. Consistency:
   DTDs enforce uniform formatting and content across XML documents, enhancing data consistency.

4. Documentation:
   DTDs serve as documentation for the structure and elements in XML, aiding understanding and development.

5. Modularity:
   DTDs enable reusable definitions, supporting modular XML design and reducing redundancy.

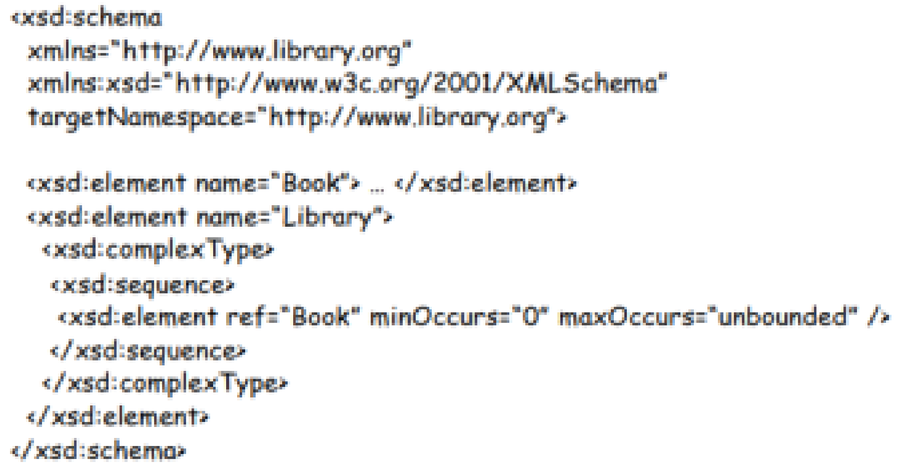
6. Efficiency:
   DTDs' simplicity leads to efficient processing and parsing of XML documents.

7. Standardization:
   DTDs provide a standardized method for specifying XML structure, reducing confusion in collaborative environments.

8. Legacy Support:
   DTDs are widely supported, making them suitable for legacy systems and older applications.

What is XML Schema?

XML Schema, often referred to as an XSD (XML Schema Definition), is a way to define the structure, content, and constraints of XML documents.
XML (Extensible Markup Language) itself is a flexible format used to represent structured data, and XML Schema provides a standardized method to describe the rules that a particular XML document must follow.



```
<xsd:schema
  xmlns="http://www.library.org"
  xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
  targetNamespace="http://www.library.org">

  <xsd:element name="Book"> ... </xsd:element>
  <xsd:element name="Library">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element ref="Book" minOccurs="0" maxOccurs="unbounded" />
     </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Key concepts related to XML Schema:

1. Structure:
   XML Schema defines the structure of an XML document by specifying the elements that can appear, their hierarchy, and the relationships between them. It helps establish a blueprint for the organization of data within an XML file.

2. Data Types:
   XML Schema allows you to define the data types of elements and attributes. This means you can specify whether an element's content should be text, numbers, dates, booleans, or other specific data types.

3. Validation:
   One of the primary purposes of an XML Schema is to validate XML documents. Validation ensures that an XML document adheres to the rules defined in the schema. This helps ensure data consistency and integrity.

4. Constraints:
   XML Schema lets you set constraints on elements and attributes. These constraints define rules that the XML document must follow. For example, you can specify that a certain element must appear a certain number of times or that a certain attribute must have a specific value.

5. Namespaces:
   XML Schemas support namespaces, which allow you to group related elements and attributes together under a unique identifier. This is particularly important when dealing with complex XML documents that might involve multiple schemas.

6. Complex Types and Elements:
   XML Schema distinguishes between simple and complex types. Complex types are used for elements that contain child elements or attributes, while simple types are used for elements with textual content.

7. Inheritance:
   XML Schema supports inheritance, allowing you to define new elements based on existing ones. This promotes code reuse and maintains consistency within a schema.

8. Documentation:
   You can include human-readable documentation within the XML Schema to help developers understand the intended usage of elements, attributes, and data types.

XML Schema is widely used in various contexts, including web services, data exchange, configuration files, and more. It plays a crucial role in ensuring interoperability and consistent data representation across different systems and applications that work with XML data.

Benefits of using XML Schema?

1. Data Validation:
   XML Schema enforces data validation, ensuring that XML documents adhere to predefined rules and constraints, thus maintaining data integrity.

2. Structured Data:
   XML Schema defines a clear structure for XML documents, facilitating better organization and understanding of data.

3. Interoperability:
   By providing a standardized format for data representation, XML Schema promotes interoperability between different systems and platforms.

4. Data Type Definition:
   XML Schema allows you to specify data types for elements and attributes, ensuring consistent handling of various data formats like text, numbers, dates, and more.

5. Constraints:
   You can set specific constraints on elements and attributes, helping to prevent invalid or inconsistent data from being included in XML documents.

6. Documentation:
   XML Schema supports inline documentation, making it easier for developers to understand the purpose and usage of different elements and attributes.

7. Namespace Support:
   XML Schema supports namespaces, enabling the categorization of elements and attributes, particularly important in complex XML documents.

8. Code Generation:
   XML Schema can be used to generate code and classes in various programming languages, streamlining the process of working with XML data in code.

9. Data Transformation:
   XML documents validated against a schema can be confidently transformed into other formats, as they adhere to a consistent structure and rules

10. Extensibility:
    XML Schema supports extensibility, allowing you to build upon existing schemas to accommodate new requirements while maintaining compatibility.

11. Standardization:
    XML Schema is an industry-standard specification, ensuring that your XML documents adhere to widely accepted practices.

12. Maintenance:
    With a well-defined XML Schema, maintenance becomes easier as changes and updates to data structures can be systematically managed.

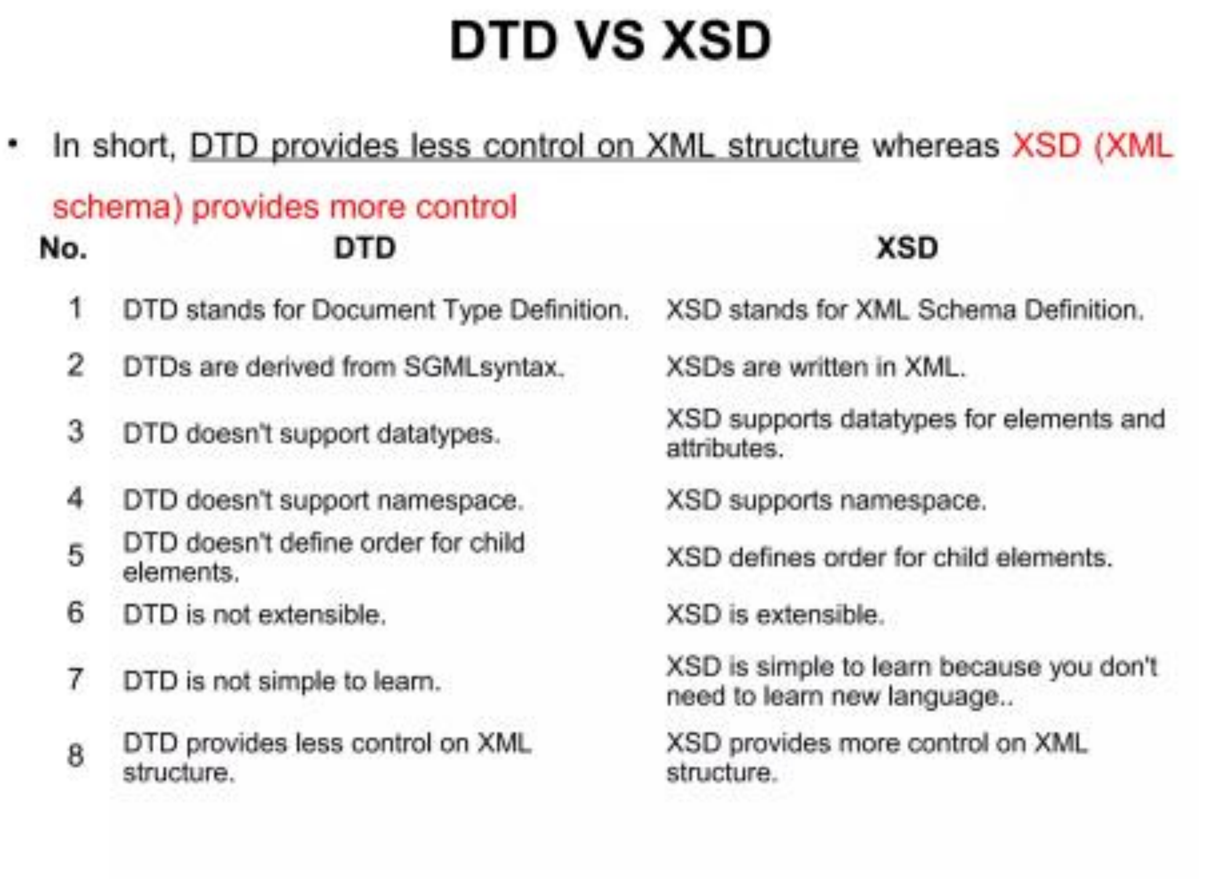
13. Error Detection:
    XML documents validated against a schema can help catch errors early in the development process, reducing debugging efforts later.

14. Schema Reuse:
    Schemas can be reused across multiple projects, promoting consistency and reducing redundant schema development.

15. Data Transformation:
    XML documents validated against a schema can be confidently transformed into other formats, as they adhere to a consistent structure and rules.

16. Data Exchange:
    XML documents conforming to a standardized schema are ideal for data exchange between different applications and organizations.

DTD vs Schema

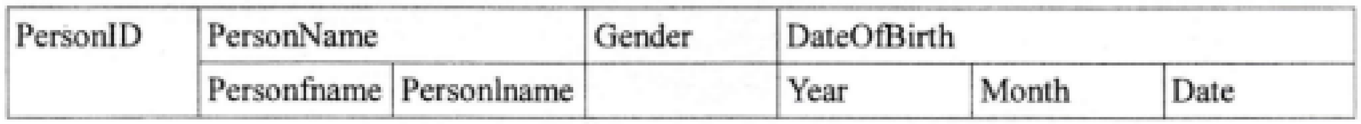


DTD VS XSD

- In short, DTD provides less control on XML structure whereas XSD (XML schema) provides more control

No.	DTD	XSD
1	DTD stands for Document Type Definition.	XSD stands for XML Schema Definition.
2	DTDs are derived from SGMLsyntax.	XSDs are written in XML.
3	DTD doesn't support datatypes.	XSD supports datatypes for elements and attributes.
4	DTD doesn't support namespace.	XSD supports namespace.
5	DTD doesn't define order for child elements.	XSD defines order for child elements.
6	DTD is not extensible.	XSD is extensible.
7	DTD is not simple to learn.	XSD is simple to learn because you don't need to learn new language..
8	DTD provides less control on XML structure.	XSD provides more control on XML structure.

Exercise 01
Write an XML code based on the following table:



PersonID	PersonName		Gender	DateOfBirth		
	Personfname	Personlname		Year	Month	Date

Exercise 02
Create an Internal DTD for a Simple Address Book

```
<addressbook>
  <contact>
    <name>John Doe</name>
    <email>john@example.com</email>
    <phone>123-456-7890</phone>
  </contact>
</addressbook>
```

Exercise 03
Create an Internal DTD for a Simple Library Catalog

```
<library>
  <book>
    <title>Harry Potter and the Sorcerer's Stone</title>
    <author>J.K. Rowling</author>
    <year>1997</year>
  </book>
</library>
```

Exercise 04
Design an External DTD for a Simple Recipe Collection. Create an external DTD file named "recipes.dtd" and reference it in the XML document.

```
<recipes>
  <recipe>
    <title>Chocolate Chip Cookies</title>
    <ingredients>Flour, sugar, chocolate chips, butter</ingredients>
    <instructions>
      1. Preheat oven...
    </instructions>
  </recipe>
</recipes>
```

Exercise 05
Design an External DTD for a Simple Product Catalog. Create an external DTD file named "products.dtd" and reference it in the XML document.

```
<products>
  <product id="p001">
    <name>Laptop</name>
    <price>$999</price>
  </product>
</products>
```

You need to submit to the LMS.

Deadline: 8th of July
File Naming Convention: SOAWS_Lab2_<IndexNumber>.pdf