2019

CI6206 Internet Programming

XML

Wong Twee Wee

Ver1.1

TOPICS

- Introduction to XML-related technologies.
- Benefits of XML.
- XML basics, well-formed XML.
- Valid XML.

Evolution of XML

- Due to the complexity of SGML, the e<u>X</u>tensible <u>M</u>arkup <u>L</u>anguage (XML) was created to be a universal format for data.
  - It is a *<u>subset</u>* of SGML with as much of the complexity removed as possible.
  - Note that an XML document is a SGML document, but SGML document may not be an XML document.

WHAT IS XML?

- Nothing more than just a text file
- A format for representing in a flexible and structured manner
- Consider the following exam results of a student :

Year	Module_Name	Module_Code	Grade
1	**INETPROG**	**CS6206**	B
1	**ENTAPPDEV**	**CI6225**	C
2	**MobileApp**	**CS7206**	A

  - Can you convert the above table to be represented by XML?

XML – Exam results

```
<?xml version="1.0" encoding="UTF-8"?>
<EXAM>

  <Module Year="1" Name="INETPROG" code="CS6206">
  <Grade>B</Grade>
  </Module>

  <Module Year="1" Name="NTAPPDEV" code="CI6225">
  <Grade>C</Grade>
  </Module>

  <Module Year="2" Name="MobileApp" code="CS7206">
  <Grade>A</Grade>
  </Module>

</EXAM>
```

Why is XML so useful?

- Human readable. Hierarchical representation of data.
- XML is compatible with the Web.
- XML structure and syntax can be validated using XML Schema and easily transformed to any other format types using transformation (XSL-T).
- Tools required to read and process XML documents are freely available on almost every major computing platforms.
- A very useful technology for *data exchange* between
  - human and machine
  - Machine and machine

XML Technologies

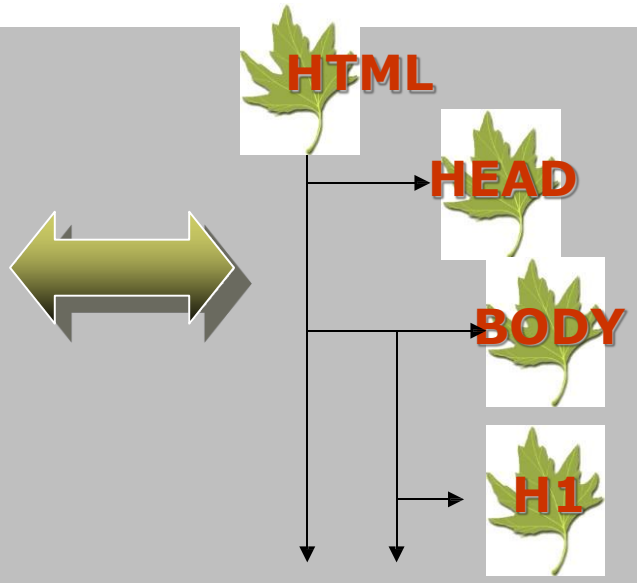
- XML is considered a bundle of technologies.
- Each set of technologies has its specific usage.
  - ***Well-formed XML***. Fundamental to XML technologies. Specifies the format and rules of an XML document.
  - ***XSL-T***. Technology for using scripts to transform XML document into another form or format. Example
  - ***DOM***. Allows manipulations of XML document by computers/software applications. Example
  - ***Schemas and DTDs.*** For ensuring that XML documents has a specified structure and set of elements. Read More

Well-Formed XML

- The most fundamental thing to know about XML is what makes up a *well-formed XML document*.
- Well-formed XML documents are made up of 2 basic things
  - Elements.
  - Hierarchical relationships of the elements.

XML HIERARCHY STRUCTURES

```
<HTML>
<HEAD/>
<BODY>
<H1>Hello</H1>
</BODY>
</HTML>
```



An XML document

Tree view of the document

- Examples of *Parent/Child* relationship
  - `HTML/HEAD, HTML/BODY, BODY/H1`
- Example of *Sibling/Sibling* relationship
  - `HEAD/BODY`

Well-Formed XML Document

- All XML documents **MUST** be well-formed.
- In order to be a well-formed XML document, certain rules need to be followed.
  - Every start-tag MUST have an end-tag.
  - Tags CANNOT overlap.
  - Each XML document can only has ONE root element.
  - Element names must obey XML naming convention.
  - XML is case-sensitive.
  - XML will keep white-space in your text (PCDATA).

XML NAMING CONVENTION

- Names for elements must follow XML naming convention.
  - Names can start with letters or the "_" characters. Numbers or punctuation characters not allowed.
  - After the first characters, numbers, "_" and "." are allowed.
  - Names CANNOT contains spaces.
  - Names cannot contain ":" character.
  - Names cannot start with the "xml" character. Even using different cases (upper, lower, mixed) are not allowed.
  - No space allowed after the "<" character and the name of the element.

EVERY START-TAG MUST HAVE AN END-TAG

```
<HTML>
   <BODY>
         <P>Hello World<BR>
         How are you
   </BODY>
</HTML>
```

No Ending Tag

No Ending Tag

Not well-formed

```
<HTML>
   <BODY>
         <P>Hello
   World</P><BR/>
         How are you
   </BODY>
</HTML>
```

Well-formed

TAGS CANNOT OVERLAP

```
<HTML>
    <BODY>
            <Font>
            <P>Hello</Font>
    World</P>
    </BODY>
</HTML>
```

Not well-formed

Font and P elements overlaps

```
<HTML>
    <BODY>
            <Font>
            <P>Hello World</P>
    </Font>
    </BODY>
</HTML>
```

Well-formed

ONLY ONE ROOT ELEMENT ALLOWED

```
<Root1>
  <Child>Hello World
  </Child>
</Root1>
<Root2>
  <Child/>
</Root2>
```

NOT Well-formed

Two roots not allowed

```
<Root1>
  <Child>Hello World
  </Child>
</Root1>
```

Well-formed

ELEMENT NAMES MUST FOLLOW NAMING CONVENTION

- The following names are not allowed

<123> or <XML123>

Invalid. Starts with Numbers or 'XML'

< My Name>

Invalid. Element names cannot contain space

<My:Name>

Invalid. Element names cannot contain ";" char

CASE SENSITIVITY

- XML documents are case sensitive. For example
  - <HTML> and <html> are different elements!

```
<HTML>
  <BODY>Hello World</body>
</HTML>
```

WHITE-SPACES IN XML

- White spaces in text content or PCDATA of an element will be retained (this is not the case in HTML).
- For example, *in HTML*, extra white spaces in text are removed.

HTML Code

```
Hello World        <BODY>
                   Many white space to
                   the left
                   </BODY>
```

Rendered Output

Hello World Many white space to the left

COMMENTS

- To add a comment in an XML document, starts the comment with the characters "<!--" and ends the comment with "-->" characters.
- Example

```
<Student>
    <!-- This is a comment -->
</Student>
```

COMMENTS (*CONTINUED…*)

- Comments cannot be inserted inside a tag. For example the following is NOT valid.

```
<Student <!- a comment --> />
```

- Comments also cannot contain the "--" characters. For example, the following is NOT valid.
- `<!- an -- invalid comment -->`

Not allowed to have "--" characters in a comment

XML DECLARATION

- XML files are text files. It is nice if we can label a file as an XML file. At the same time, we can provide more information about the XML file. This is done using the *XML declaration*.
- A typical XML declaration is put at the very <u>beginning</u> of an XML file and looks like the following:

```
<?xml version ='1.0' encoding='UTF-16' standalone='yes' ?>
<RootElement>
    …
</RootElement>
```

XML DECLARATION (*CONTINUED…*)

- Things to note about XML declarations:
  - Start with the characters "`<?xml`" and ends with "`?>`"
  - The *version* attribute is mandatory (must include).
    - Current version is at *1.0*.
    - If you specify a version > 1.0, the version 1.0 parsers is suppose to reject the XML document.
  - The *encoding* and *standalone* attributes are optional (see later slides).

USING ILLEGAL PCDATA CHARACTERS

- There are some characters that are reserved and are not allowed within a PCDATA.
- If you really need to use them, there are two methods
  - Use entity references. Replace the characters like `&` and `<` with the corresponding entity reference. In other words, replace
    - `&` with `&`
    - `>` with `>`
    - `<` with `<`
    - `'` with `'`
    - `"` with `"`

USING ILLEGAL PCDATA CHARACTERS (*CONTINUED…*)

- The second method is to use Character DATA (*CDATA*) section.
  - If you have many illegal characters in your PCDATA, the XML document can quickly become unreadable. For example

```
<ATag>if (6 &lt; 7) then b = &quot;&apos;Hello&apos;&quot;</Atag>
```

  - In such cases we can enclose the PCDATA of our element with CDATA section. A CDATA section starts with "`![CDATA[`" and ends with "`]]`"

```
<ATag>![CDATA[if (6 is > 7) then b = "'Hello'"</ATag>]]
```

VALID XML

- With well-formed XML document, we are certain that the document follows a strict XML syntax. But this is <u>not sufficient</u>.
- One major use of XML document is for *machine-machine communication*. This requires that we have a way to decide and specify document structure.
- Hence, we need to ensure that the XML document that all parties are using are of the same *type* of XML document.

VALID XML

Where is the name of the book?

```
<Library>
<Book name="ABC">
</Book>
</Library>
```

```
<Library>
<Book>
<Name>ABC</Name>
</Book>
</Library>
```

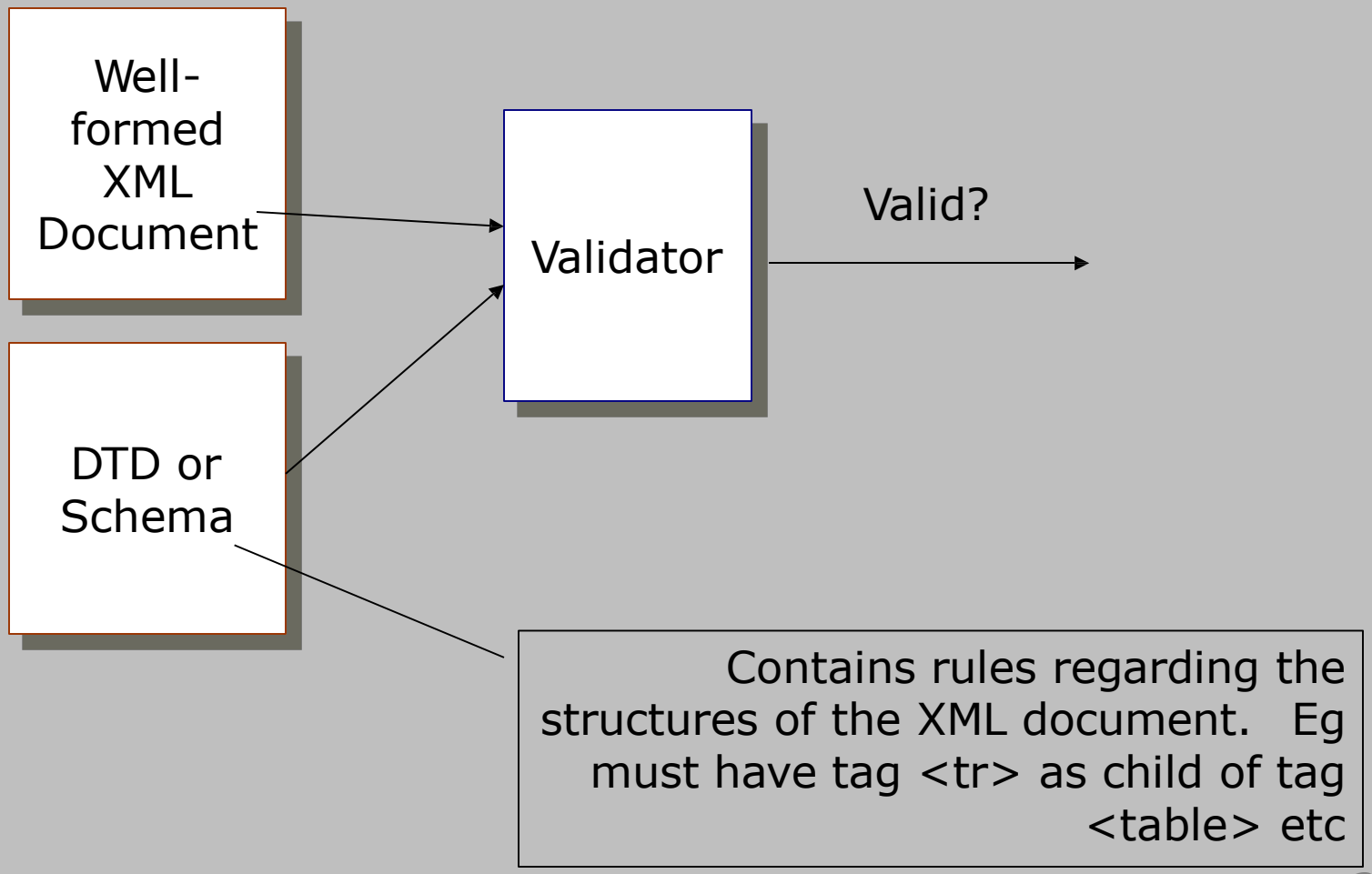
VALID XML

- We need to
  - Describe and define document structure.
  - Communicate the defined document structure.
  - Check required elements are present.
  - Check that disallowed element are NOT present.
  - Enforce element content, tree structure and element attribute values.
  - Provides default values for unspecified attribute values
  - Use standard document formats and data structures.
- *Document Type Definitions (DTDs)* and *XML schemas* are used for this purpose.

Valid XML

- DTD and Schema allows users to specify additional rules that an XML document needs to satisfy.
- Well-formed rules are pre-defined by W3C. Rules in DTDs and Schemas are defined by users.
- Example, in DTD and schema we are interested in laying down rules like
  - "A TABLE element can only have TR element as a child"
  - "A TABLE element can have an attribute called border"
  - "HTML element can only have HEAD or BODY as child elements"

VALID XML


Well-formed XML Document
DTD or Schema
Validator
Valid?
Contains rules regarding the structures of the XML document. Eg must have tag <tr> as child of tag <table> etc

VALID XML

- Example – XHTML

```
<?xml version="1.0" encoding="windows-1252"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en"
   xml:lang="en">
   <head>
        <title>Hello World</title>
   </head>
   <body>
        <tr>Hello World 2</tr>
   </body>
</html>
```

The above HTML codes is a well-formed XML but is it a valid XHTML document?

SUMMARY

- After this lecture, you should understand the followings:
  - What are the benefits of using XML.
  - What is a well-formed XML document.
  - What is a valid XML document.
  - Online XML Validator from w3schools
  - Online XML Validator (File upload) from w3
  - XML e-learning resource