CACS-205
Web Technology
(BCA, TU)

Ganesh Khatri
kh6ganesh@gmail.com

Chapter 3 : Client Tier

- Introduction to XML
- Elements and Attributes
- Rules for writing XML
- Namespaces
- Schema : Simple and Complex types
- XSD Attributes
- Facets
- DTD
- XSL / XSLT
- Xpath
- Xquery
- Creating XML Parser

XML Introduction

- XML stands for eXtensible Markup Language.
- XML was designed to store and transport data.
- XML was designed to be both human- and machine-readable
- XML is a software- and hardware-independent tool for storing and transporting data.
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML Does Not DO Anything

- Maybe it is a little hard to understand, but XML does not DO anything.
- This note is a note to Tove from Jani, stored as XML

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

- The XML above is quite self-descriptive:
  - It has sender information.
  - It has receiver information
  - It has a heading
  - It has a message body

XML Does Not DO Anything

- But still, the XML does not DO anything.
- XML is just information wrapped in tags
- Someone must write a piece of software to send, receive, store, or display it:

The Difference Between XML and HTML

- XML and HTML were designed with different goals:
  - XML was designed to carry data - with focus on what data is
  - HTML was designed to display data - with focus on how data looks
  - XML tags are not predefined like HTML tags are

XML Does Not Use Predefined Tags

- The XML language has no predefined tags.
- The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.
- HTML works with predefined tags like <p>, <h1>, <table>, etc.
- With XML, the author must define both the tags and the document structure

XML is Extensible

- Most XML applications will work as expected even if new data is added (or removed).
- Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <body>).
- Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.
- The way XML is constructed, older version of the application can still work:

```
<note>
  <date>2015-09-01</date>
  <hour>08:30</hour>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>
```

XML Simplifies Things

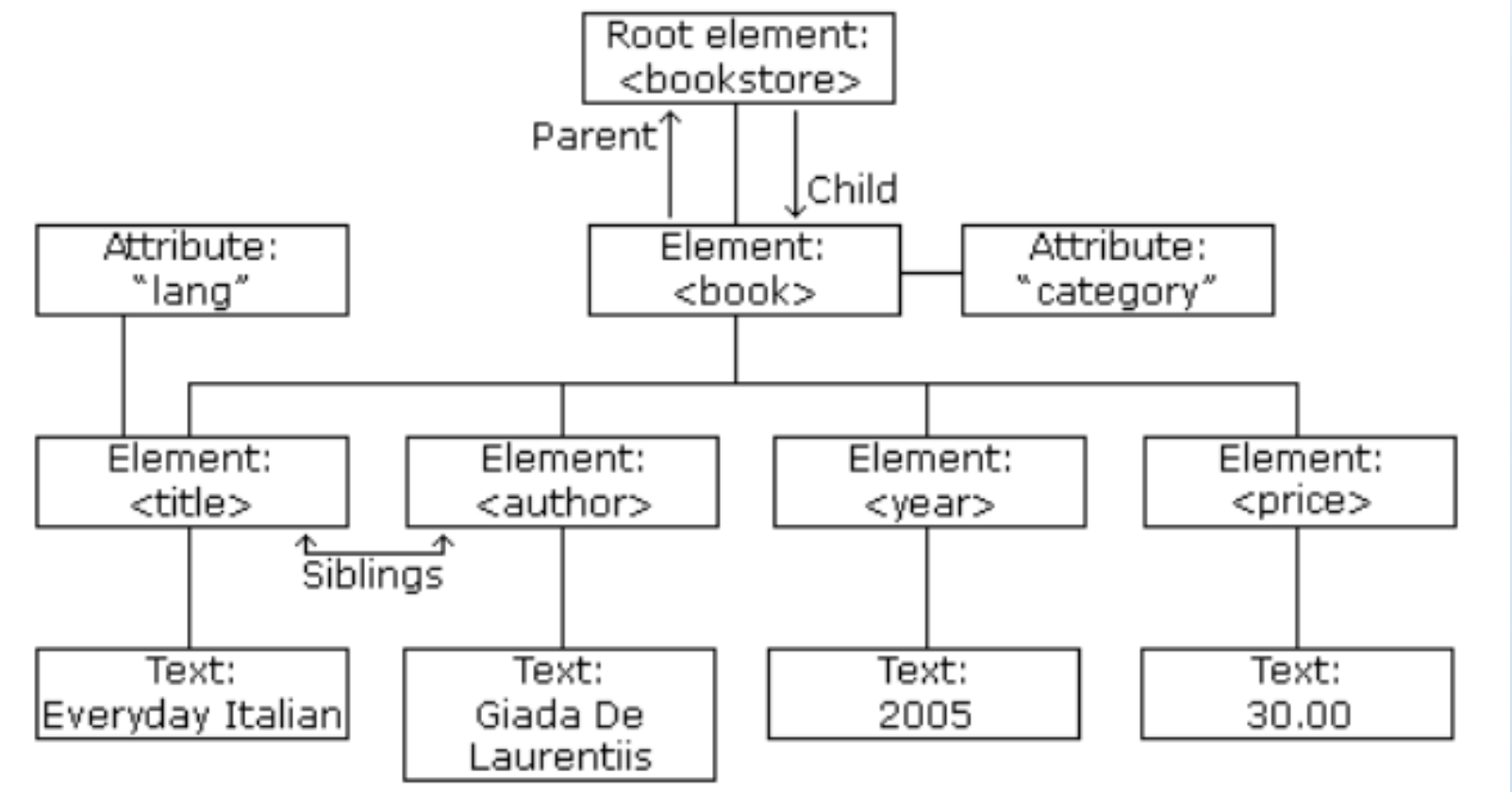
- XML
  - simplifies data sharing
  - simplifies data transport
  - simplifies platform changes
  - simplifies data availability
- Many computer systems contain data in incompatible formats. Exchanging data between incompatible systems (or upgraded systems) is a time-consuming task for web developers. Large amounts of data must be converted, and incompatible data is often lost
- XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data
- XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data
- With XML, data can be available to all kinds of "reading machines" like people, computers, voice machines, news feeds, etc

How Can XML be Used?

- XML is used in many aspects of web development.
- XML is often used to separate data from presentation
- XML Separates Data from Presentation
- XML is Often a Complement to HTML
- XML Separates Data from HTML

XML Tree

- XML documents form a tree structure that starts at "the root" and branches to "the leaves"



An Example XML Document

- The image above represents books in this XML:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML Syntax Rules

- The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.
  - XML Documents Must Have a Root Element
  - The XML Prolog
  - All XML Elements Must Have a Closing Tag
  - XML Tags are Case Sensitive
  - XML Elements Must be Properly Nested
  - XML Attribute Values Must Always be Quoted
  - Entity References
  - Comments in XML
  - White-space is Preserved in XML

XML Documents Must Have a Root Element

- XML documents must contain one root element that is the parent of all other elements:
- In this example <note> is the root element:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

The XML Prolog

- This line is called the XML prolog:

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

- The XML prolog is optional. If it exists, it must come first in the document.
- XML documents can contain international characters, like Norwegian øæå or French êèé.
- To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.
- UTF-8 is the default character encoding for XML documents
- UTF-8 is also the default encoding for HTML5, CSS, JavaScript, PHP, and SQL.

All XML Elements Must Have a Closing Tag

- In XML, it is illegal to omit the closing tag. All elements must have a closing tag:

```
<p>This is a paragraph.</p>
<br />
```

- **Note:** The XML prolog does not have a closing tag! This is not an error. The prolog is not a part of the XML document.

XML Tags are Case Sensitive

- XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.
- Opening and closing tags must be written with the same case:

```
<message>This is correct</message>
```

- "Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing

XML Elements Must be Properly Nested

- In HTML, you might see improperly nested elements:

```
<b><i>This text is bold and italic</b></i>
```

- In XML, all elements must be properly nested within each other:

```
<b><i>This text is bold and italic</i></b>
```

- In the example above, "Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

XML Attribute Values Must Always be Quoted

- XML elements can have attributes in name/value pairs just like in HTML.
- In XML, the attribute values must always be quoted

```
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

Entity References

- Some characters have a special meaning in XML.
- If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.
- This will generate an XML error:

```
<message>salary < 1000</message>
```

- To avoid this error, replace the "<" character with an **entity reference**:

```
<message>salary &lt; 1000</message>
```

Entity References

- There are 5 pre-defined entity references in XML:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

- Only < and & are strictly illegal in XML, but it is a good habit to replace > with > as well.

Comments in XML

- The syntax for writing comments in XML is similar to that of HTML:

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

- Two dashes in the middle of a comment are not allowed:

```
<!-- This is an invalid -- comment -->
```

White-space is Preserved in XML

- XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space):

XML:	Hello Tove
HTML:	Hello Tove

XML Elements

- An XML document contains XML Elements.
- An XML element is everything from (including) the element's start tag to (including) the element's end tag.

```
<price>29.99</price>
```

- An element can contain:
  - text
  - attributes
  - other elements
  - or a mix of the above

```
<bookstore>
  <book category="children">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Empty XML Elements

- An element with no content is said to be empty.
- In XML, you can indicate an empty element like this

```
<element></element>
```

- You can also use a so called self-closing tag:

```
<element />
```

- The two forms produce identical results in XML software (Readers, Parsers, Browsers).

XML Naming Rules

- XML elements must follow these naming rules:
  - Element names are case-sensitive
  - Element names must start with a letter or underscore
  - Element names cannot start with the letters xml (or XML, or Xml, etc)
  - Element names can contain letters, digits, hyphens, underscores, and periods
  - Element names cannot contain spaces
- Any name can be used, no words are reserved (except xml).

Naming Styles

- There are no naming styles defined for XML elements. But here are some commonly used:

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first

- If you choose a naming style, it is good to be consistent!
- XML documents often have a corresponding database. A common practice is to use the naming rules of the database for the XML elements
- Camel case is a common naming rule in JavaScript.

XML Attributes

- XML elements can have attributes, just like HTML.
- Attributes are designed to contain data related to a specific element
- XML Attributes Must be Quoted
- `<person gender="female">` or `<person gender='female'>`
- If the attribute value itself contains double quotes you can use single quotes, like in this example:

```
<gangster name='George "Shotgun" Ziegler'>
```

- Or you can use character entities

```
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

XML Elements vs. Attributes

- In the first example gender is an attribute.
- In the last, gender is an element. Both examples provide the same information.
- There are no rules about when to use attributes or when to use elements in XML

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

XML Namespaces

- XML Namespaces provide a method to avoid element name conflicts.
- In XML, element names are defined by the developer.
- This often results in a conflict when trying to mix XML documents from different XML applications
- This XML carries HTML table information:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

XML Namespaces

- This XML carries information about a table (a piece of furniture):

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

- If these XML fragments were added together, there would be a name conflict.
- Both contain a <table> element, but the elements have different content and meaning
- A user or an XML application will not know how to handle these differences.

Solving the Name Conflict Using a Prefix

- Name conflicts in XML can easily be avoided using a name prefix.
- This XML carries information about an HTML table, and a piece of furniture:
- In this example, there will be no conflict because the two <table> elements have different names.

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

XML Namespaces - The xmlns Attribute

- When using prefixes in XML, a namespace for the prefix must be defined.
- The namespace can be defined by an **xmlns** attribute in the start tag of an element.
- The namespace declaration has the following syntax. xmlns:*prefix*="*URI*"

```
<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="https://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

XML Namespaces - The xmlns Attribute

- **Note:** The namespace URI is not used by the parser to look up information.
- The purpose of using an URI is to give the namespace a unique name.
- However, companies often use the namespace as a pointer to a web page containing namespace information.

Uniform Resource Identifier (URI)

- A **Uniform Resource Identifier** (URI) is a string of characters which identifies an Internet Resource.
- The most common URI is the **Uniform Resource Locator** (URL) which identifies an Internet domain address. Another, not so common type of URI is the **Uniform Resource Name** (URN).

Default Namespaces

- Defining a default namespace for an element saves us from using prefixes in all the child elements.
- It has the following syntax:

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```