Xml

XML is a markup language designed to store and transport data. It is supposed to be both human readable and machine readable.

Unlike HTML, XML does not specify how to display or to format data, it just carries it.

Distinctions are made between the **content** and the **markup**. In short, content could be anything, markup is defined.

Some definitions and introductions

XML Documents are basically made up of *elements* which can have *attributes* describing them and may contain some textual content or more elements as its children. All XML documents must have a root element, which is the ancestor of all the other elements in the document.

XML Parsers are designed to be very strict, and will stop parsing malformed documents. Therefore it must be ensured that all XML documents follow the XML Syntax Rules.

```
<!-- This is a comment. It must not contain two consecutive hyphens (-). -->
<!-- Comments can span
  multiple lines -->

<!-- Elements -->
<!-- An element is a basic XML component. There are two types, empty: -->
<element1 attribute="value" /> <!-- Empty elements do not hold any content -->
<!-- and non-empty: -->
<element2 attribute="value">Content</element2>
<!-- Element names may only contain alphabets and numbers. -->

<empty /> <!-- An element either consists an empty element tag… -->
<!-- …which does not hold any content and is pure markup. -->

<notempty> <!-- Or, it consists of a start tag… -->
  <!-- …some content… -->
</notempty> <!-- and an end tag. -->

<!-- Element names are case sensitive. -->
<element />
<!-- is not the same as -->
<eLEMENT />

<!-- Attributes -->
<!-- An attribute is a key-value pair and exists within an element. -->
<element attribute="value" another="anotherValue" many="space-separated list" />
<!-- An attribute may appear only once in an element. It holds just one value.
  Common workarounds to this involve the use of space-separated lists. -->

<!-- Nesting elements -->
<!-- An element's content may include other elements: -->
<parent>
  <child>Text</child>
  <emptysibling />
</parent>
<!-- Standard tree nomenclature is followed. Each element being called a node.
```

```
  An ancestor a level up is the parent, descendants a level down are children.
  Elements within the same parent element are siblings. -->

<!-- XML preserves whitespace. -->
<child>
  Text
</child>
<!-- is not the same as -->
<child>Text</child>
```

An XML document

This is what makes XML versatile. It is human readable too. The following document tells us that it defines a bookstore which sells three books, one of which is Learning XML by Erik T. Ray. All this without having used an XML Parser yet.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is called an XML prolog. Optional, but recommended. -->
<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>
```

Well-formedness and Validation

A XML document is *well-formed* if it is syntactically correct. However, it is possible to add more constraints to the document, using Document Type Definitions (DTDs). A document whose elements are attributes are declared in a DTD and which follows the grammar specified in that DTD is called *valid* with respect to that DTD, in addition to being well-formed.

```
<!-- Declaring a DTD externally: -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bookstore SYSTEM "Bookstore.dtd">
<!-- Declares that bookstore is our root element and 'Bookstore.dtd' is the path
  to our DTD file. -->
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
```

```
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>

<!-- The DTD file: -->
<!ELEMENT bookstore (book+)>
<!-- The bookstore element may contain one or more child book elements. -->
<!ELEMENT book (title, price)>
<!-- Each book must have a title and a price as its children. -->
<!ATTLIST book category CDATA "Literature">
<!-- A book should have a category attribute. If it doesn't, its default value
  will be 'Literature'. -->
<!ELEMENT title (#PCDATA)>
<!-- The element title must only contain parsed character data. That is, it may
  only contain text which is read by the parser and must not contain children.
  Compare with CDATA, or character data. -->
<!ELEMENT price (#PCDATA)>
]>

<!-- The DTD could be declared inside the XML file itself.-->

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

<!DOCTYPE bookstore [
<!ELEMENT bookstore (book+)>
<!ELEMENT book (title, price)>
<!ATTLIST book category CDATA "Literature">
<!ELEMENT title (#PCDATA)>
<!ELEMENT price (#PCDATA)>
]>

<bookstore>
  <book category="COOKING">
    <title>Everyday Italian</title>
    <price>30.00</price>
  </book>
</bookstore>
```

DTD Compatibility and XML Schema Definitions

Support for DTDs is ubiquitous because they are so old. Unfortunately, modern XML features like namespaces are not supported by DTDs. XML Schema Definitions (XSDs) are meant to replace DTDs for defining XML document grammar.

Resources

- Validate your XML

Further Reading

- XML Schema Definitions Tutorial
- DTD Tutorial
- XML Tutorial
- Using XPath queries to parse XML