XML – eXtensible Markup Language

What is xml?

- Xml (eXtensible Markup Language) is a mark up language.
- XML is designed to store and transport data.
- Xml was released in late 90's. it was created to provide an easy to use and store self describing data.
- XML became a W3C Recommendation on February 10, 1998.
- XML is designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.

Advantages of XML:

- XML separates data from HTML
- XML simplifies data sharing
- XML simplifies data transport
- XML simplifies platform changes
- XML simplifies data availability

Example:

XML documents create a hierarchical structure looks like a tree so it is known as XML Tree that starts at "the root" and branches to "the leaves".

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<note>

  <to>Tove</to>

  <from>Jani</from>

  <heading>Reminder</heading>

  <body>Don't forget me this weekend!</body>

</note>
```

The first line is the XML declaration. It defines the XML version (1.0) and the encoding used (ISO-8859-1 = Latin-1/West European character set).

The next line describes the root element of the document (like saying: "this document is a note"):

The next 4 lines describe 4 child elements of the root (to, from, heading, and body).

XML documents must contain a root element. This element is "the parent" of all other elements.

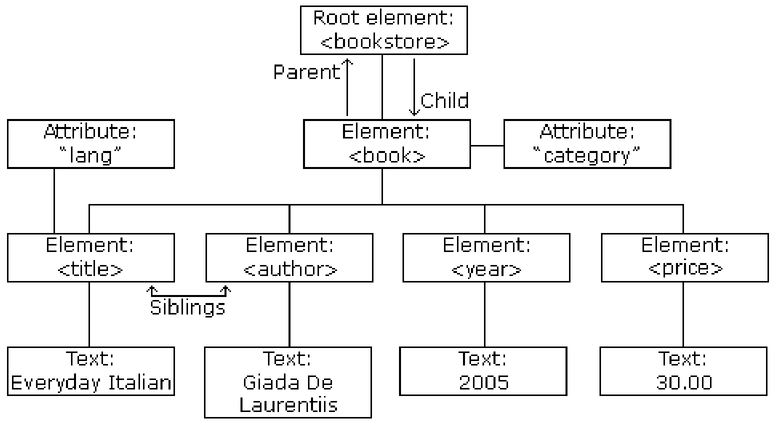
The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree.

XML Syntax: All elements can have sub elements (child elements).

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

Example-1:

```
<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>
```



The root element in the example is <bookstore>. All elements in the document are contained within <bookstore>.

The <book> element has 4 children: <title>,< author>, <year> and <price>.

Transaction Data

Thousands of XML formats exist, in many different industries, to describe day-to-day data transactions:

- Stocks and Shares

- Financial transactions
- Medical data
- Mathematical data
- Scientific measurements
- News information
- Weather services

XML related Technology:

No.	Technology	Meaning	Description
1)	XHTML	Extensible html	It is a clearer and stricter version of XML. It belongs to the family of XML markup languages. It was developed to make html more extensible and increase inter-operability with other data.
2)	XML DOM	XML document object model	It is a standard document model that is used to access and manipulate XML. It defines the XML file in tree structure.
3)	XSL it contain three parts: i) XSLT (xsl transform) ii) XSL iii)XPath	Extensible style sheet language	i) It transforms XML into other formats, like html. ii) It is used for formatting XML to screen, paper etc. iii) It is a language to navigate XML documents.
4)	XQuery	XML query language	It is a XML based language which is used to query XML based data.
5)	DTD	Document type definition	It is an standard which is used to define the legal elements in an XML document.
6)	XSD	XML schema definition	It is an XML based alternative to dtd. It is used to describe the structure of an XML document.

Document Type Definition (DTD) :

- DTD stands for Document Type Definition.
- It is a document which defines the structure of an XML document.
- It is used to describe the attributes of XML language precisely.
- It can be classified into two types namely internal DTD and external DTD which can be specified inside a document or outside a document.
- DTD mainly checks the grammar and validity of a XML document. It checks that a XML document has a valid structure or not.

Example: An Internal DTD Declaration

If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition:

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

Above DTD is interpreted like this:

!DOCTYPE note defines that the root element of this document is note

!ELEMENT note defines that the note element must contain four elements: "to,from,heading,body"

!ELEMENT to defines that to element is "#PCDATA" type. (parse-able data type)

XML DTD with entity declaration

A doctype declaration can also define special strings that can be used in the XML file.

An entity has three parts:

- An ampersand (&)
- An entity name
- A semicolon (;)

Syntax to declare entity: <!ENTITY entity-name "entity-value">

author.xml

```
<?xml version="1.0" standalone="yes" ?>
```

```
<!DOCTYPE author [

  <!ELEMENT author (#PCDATA)>

  <!ENTITY sj "Sonoo Jaiswal">

]>

<author>&sj;</author>
```

Example: An External DTD Declaration

employee.xml	**employee.dtd**

`<?xml version="1.0"?>`
`<!DOCTYPE employee SYSTEM "employee.dtd">`
`<employee>`
`  <firstname>ABC</firstname>`
`  <lastname>PATEL</lastname>`
`  <email>abp01@gnu.ac.in</email>`
`</employee>`
	`<!ELEMENT employee (firstname,lastname,email)>`
	`<!ELEMENT firstname (#PCDATA)>`
	`<!ELEMENT lastname (#PCDATA)>`
	`<!ELEMENT email (#PCDATA)>`

2. XML Schema Definition (XSD) :

- XSD stands for XML Schema Definition.
- It is a way to describe the structure of a XML document.
- It defines the rules for all the attributes and elements in a XML document.
- It doesn't require processing by a parser. XSD checks for the correctness of the structure of the XML file.
- XSD was first published in 2001 and after that it was published in 2004.

XML Schema Data types

There are two types of data types in XML schema.

simpleType
complexType

simpleType

The simpleType allows you to have text-based elements. It contains less attributes, child elements, and cannot be left empty.

complexType

The complexType allows you to hold multiple attributes and elements. It can contain additional sub elements and can be left empty.

Example:

employee.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
<xs:element name="employee">
  <xs:complexType>
```

```
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
  </xs:schema>
```

employee.xml

```
<?xml version="1.0"?>
<employee xsi:schemaLocation=
 ”employee.xsd">
<firstname>vimal</firstname>
  <lastname>jaiswal</lastname>
  <email>vimal@javatpoint.com</email>
</employee>
```

DTD	XSD
DTD are the declarations that define a document type for SGML.	XSD describes the elements in a XML document.
It doesn’t support namespace.	It supports namespace.
It is comparatively harder than XSD.	It is relatively simpler than DTD.
It doesn’t support data types.	It supports data types.
SGML syntax is used for DTD.	XML is used for writing XSD.
It is not extensible in nature.	It is extensible in nature.
It doesn’t give us much control on structure of XML document.	It gives us more control on structure of XML document.

Parse XML in PHP
PHP SimpleXML Parser:
→SimpleXML is a tree-based parser.
→SimpleXML provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.

PHP SimpleXML - Read From String

The PHP `simplexml_load_string()` function is used to read XML data from a string.

Example:

```
<?php

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

<note>

<to>Komal</to>
```

```
<from>Meena</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>";

$xml=simplexml_load_string($myXMLData) or die("Error: Cannot create object");

print_r($xml);

echo $xml->to . "<br>";

echo $xml->from . "<br>";

?>
```

Output:

```
SimpleXMLElement Object ( [to] => Komal [from] => Meena [heading] => Reminder
[body] => Don't forget me this weekend! ) Komal Meena
```

PHP SimpleXML - Read From File

The PHP `simplexml_load_file()` function is used to read XML data from a file.

```
<?php
$xml=simplexml_load_file("note.xml") or die("Error: Cannot create object");
print_r($xml);
?>
```

PHP SimpleXML - Get Node Values

Example:

Book.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>
</bookstore>
```

Book.php

```
<?php
$xml=simplexml_load_file("Book.xml") or die("Error: Cannot create object");
echo $xml->book[0]->title . "<br>";
echo $xml->book[1]->title;
?>
```

PHP XML DOM Parser

The DOM parser is a tree-based parser.
Look at the following XML document fraction:

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

The DOM sees the XML above as a tree structure:

Level 1: XML Document
Level 2: Root element: <from>
Level 3: Text element: "Jani"

Example:
ab.php

```
<?php
//Read XML by creating DOM object
$xmlDoc = new DOMDocument();
$xmlDoc->load("ab.xml");
print $xmlDoc->saveXML();
?>
```

The example above creates a DOMDocument-Object and loads the XML from "note.xml" into it.Then the saveXML() function puts the internal XML document into a string, so we can output it.

ab.xml

```
<?xml version="1.0"?>
<Company>
  <Employee>
     <FirstName>Tanmay</FirstName>
     <LastName>Patil</LastName>
     <ContactNo>1234567890</ContactNo>
     <Email>tanmaypatil@xyz.com</Email>
     <Address>
         <City>Bangalore</City>
         <State>Karnataka</State>
         <Zip>560212</Zip>
     </Address>
  </Employee>
</Company>
```

Output: Tanmay Patil 1234567890 tanmaypatil@xyz.com

Bangalore Karnataka 560212