XML

Advantages and Disadvantages of XML

In the last tutorial we discussed the features of XML. In this guide, we will learn the **advantages and disadvantages of XML**.

Advantages of XML

1. XML is platform independent and programming language independent, thus it can be used on any system and supports the technology change when that happens.

2. XML supports unicode. Unicode is an international encoding standard for use with different languages and scripts, by which each letter, digit, or symbol is assigned a unique numeric value that applies across different platforms and programs. This feature allows XML to transmit any information written in any human language.

3. The data stored and transported using XML can be changed at any point of time without affecting the data presentation. Generally other markup language such as HTML is used for data presentation, HTML gets the data from XML and display it on the GUI (graphical user interface), once data is updated in XML, it does reflect in HTML without making any change in HTML GUI.

4. XML allows validation using DTD and Schema. This validation ensures that the XML document is free from any syntax error.

5. XML simplifies data sharing between various systems because of its platform independent nature. XML data doesn't require any conversion when transferred between different systems.

Disadvantages of XML

1. XML syntax is verbose and redundant compared to other text-based data transmission formats such as JSON.

2. The redundancy in syntax of XML causes higher storage and transportation cost when the volume of data is large.

3. XML document is less readable compared to other text-based data transmission formats such as JSON.

4. XML doesn't support array.

5. XML file sizes are usually very large due to its verbose nature, it is totally dependant on who is writing it.

XML Example – Student data

Lets take a look at the another example of XML. In this XML document we have the details of the few students. Here <students> is the root element, <student> is the child element and name, age, subject and gender are sub-child elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
   <gender>Male</gender>
 </student>
 <student>
   <name>Daryl Dixon </name>
   <age>33</age>
   <subject>Science</subject>
   <gender>Male</gender>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
   <gender>Female</gender>
 </student>
</students>
```

The first line `<?xml version="1.0" encoding="UTF-8"?>` is called **XML Prolog**. It is optional, however when we include it in the XML document, it should always be the first line of the document. XML Prolog defines the XML version and the encoding used in the XML document.

XML Tree Structure

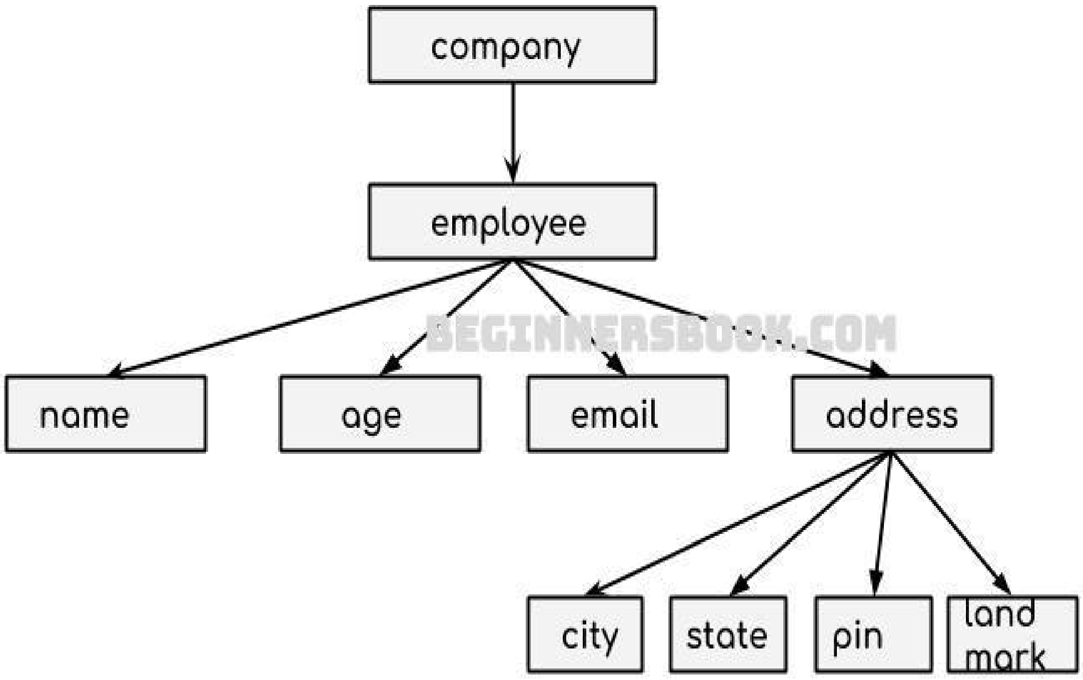
XML document has a tree structure, where the root element is at the top and the child elements are connected to the root elements, the same way, how leaves are connected to tree through branches. We will first see an example of XML document and then we will draw a tree structure based on the example.

Example of XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<company>
  <employee>
    <name>Negan</name>
    <age>40</age>
    <email>imnegan@twd.com</email>
    <address>
      <city>Noida</city>
      <state>Uttar Pradesh</state>
      <pin>201301</pin>
      <landmark>Near hill top</landmark>
    </address>
  </employee>
</company>
```

Tree Structure

The tree structure of the above XML document would look like this:

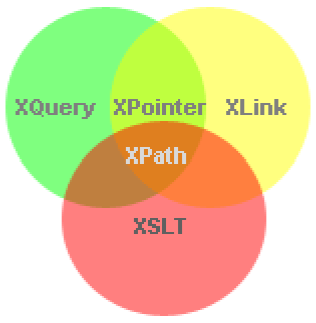


Root element: <company>
Child elements: <name>, <age>, <email> & <address>
Sub-child elements: <city>, <state>, <pin> & <landmark>

XPath

XPath is a major element in the XSLT standard.

XPath can be used to navigate through elements and attributes in an XML document.



- XPath stands for XML Path Language
- XPath uses "path like" syntax to identify and navigate nodes in an XML document
- XPath contains over 200 built-in functions
- XPath is a major element in the XSLT standard
- XPath is a W3C recommendation

XPath (XML Path Language) is a query language for selecting nodes from an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath was defined by the World Wide Web Consortium (W3C).

XPath Syntax

XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by following a path or steps.

The XML Example Document

We will use the following XML document in the examples below.

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

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

XPath - Expression

An XPath expression generally defines a pattern in order to select a set of nodes. These patterns are used by XSLT to perform transformations or by XPointer for addressing purpose.

XPath specification specifies seven types of nodes which can be the output of execution of the XPath expression.

- Root
- Element
- Text
- Attribute
- Comment
- Processing Instruction
- Namespace

XPath uses a path expression to select node or a list of nodes from an XML document.

Selecting Nodes

XPath uses path expressions to select nodes in an XML document. The node is selected by following a path or steps. The most useful path expressions are listed below:

Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document

bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

Predicates

Predicates are used to find a specific node or a node that contains a specific value.

Predicates are always embedded in square brackets.

In the table below we have listed some path expressions with predicates and the result of the expressions:

Path Expression	**Result**
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element. **Note:** In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1]. To solve this problem in IE, set the SelectionLanguage to XPath: *In JavaScript:* *xml*.setProperty("SelectionLanguage","XPath");
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='en']	Selects all the title elements that have a "lang" attribute with a value of "en"
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00

/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

Selecting Unknown Nodes

XPath wildcards can be used to select unknown XML nodes.

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
/bookstore/*	Selects all the child element nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have at least one attribute of any kind

Selecting Several Paths

By using the | operator in an XPath expression you can select several paths.

In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result

//book/title | //book/price	Selects all the title AND price elements of all book elements
//title | //price	Selects all the title AND price elements in the document
/bookstore/book/title | //price	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document